E 14

```
PPPPPPPPPPPP        AAAAAAAAA     TTTTTTTTTTTTTTT     CCCCCCCCCCCC  HHH         HHH
PPPPPPPPPPPP        AAAAAAAAA     TTTTTTTTTTTTTTT     CCCCCCCCCCCC  HHH         HHH
PPPPPPPPPPPP        AAAAAAAAA     TTTTTTTTTTTTTTT     CCCCCCCCCCCC  HHH         HHH
PPP         PPP  AAA         AAA        TTT        CCC              HHH         HHH
PPP         PPP  AAA         AAA        TTT        CCC              HHH         HHH
PPP         PPP  AAA         AAA        TTT        CCC              HHH         HHH
PPP         PPP  AAA         AAA        TTT        CCC              HHH         HHH
PPP         PPP  AAA         AAA        TTT        CCC              HHH         HHH
PPP         PPP  AAA         AAA        TTT        CCC              HHH         HHH
PPPPPPPPPPPP     AAA         AAA        TTT        CCC              HHHHHHHHHHHHHHH
PPPPPPPPPPPP     AAA         AAA        TTT        CCC              HHHHHHHHHHHHHHH
PPPPPPPPPPPP     AAA         AAA        TTT        CCC              HHHHHHHHHHHHHHH
PPP              AAAAAAAAAAAAAAA        TTT        CCC              HHH         HHH
PPP              AAAAAAAAAAAAAAA        TTT        CCC              HHH         HHH
PPP              AAAAAAAAAAAAAAA        TTT        CCC              HHH         HHH
PPP              AAA         AAA        TTT        CCC              HHH         HHH
PPP              AAA         AAA        TTT        CCC              HHH         HHH
PPP              AAA         AAA        TTT        CCC              HHH         HHH
PPP              AAA         AAA        TTT           CCCCCCCCCCCC  HHH         HHH
PPP              AAA         AAA        TTT           CCCCCCCCCCCC  HHH         HHH
PPP              AAA         AAA        TTT           CCCCCCCCCCCC  HHH         HHH
```

```
**FILE**ID**PATEXA                                  L  3
```

```
PPPPPPPP      AAAAAA    TTTTTTTTTT  EEEEEEEEEE  XX      XX    AAAAAA
PPPPPPPP      AAAAAA    TTTTTTTTTT  EEEEEEEEEE  XX      XX    AAAAAA
PP      PP  AA      AA      TT      EE          XX      XX  AA      AA
PP      PP  AA      AA      TT      EE          XX      XX  AA      AA
PP      PP  AA      AA      TT      EE            XX  XX    AA      AA
PP      PP  AA      AA      TT      EE            XX  XX    AA      AA
PPPPPPPP    AA      AA      TT      EEEEEEEE        XX      AA      AA
PPPPPPPP    AA      AA      TT      EEEEEEEE        XX      AA      AA
PP          AAAAAAAAAA      TT      EE            XX  XX    AAAAAAAAAA
PP          AAAAAAAAAA      TT      EE            XX  XX    AAAAAAAAAA
PP          AA      AA      TT      EE          XX      XX  AA      AA      ....
PP          AA      AA      TT      EE          XX      XX  AA      AA      ....
PP          AA      AA      TT      EEEEEEEEEE  XX      XX  AA      AA      ....
PP          AA      AA      TT      EEEEEEEEEE  XX      XX  AA      AA      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

PA
VO

```
MODULE PATEXA (
                %IF %VARIANT EQL 1
                %THEN
                        ADDRESSING_MODE (EXTERNAL = LONG_RELATIVE, NONEXTERNAL = LONG_RELATIVE),
                %FI
                IDENT = 'V04-000') =
BEGIN

!****************************************************************************
!*                                                                          *
!*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
!*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
!*  ALL RIGHTS RESERVED.                                                    *
!*                                                                          *
!*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
!*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
!*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
!*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
!*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
!*  TRANSFERRED.                                                            *
!*                                                                          *
!*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
!*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
!*  CORPORATION.                                                            *
!*                                                                          *
!*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
!*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
!*                                                                          *
!*                                                                          *
!****************************************************************************
!
! FACILITY:     PATCH
!
!++
! FUNCTIONAL DESCRIPTION:
!
!       EXAMINE, DEPOSIT, AND DELETE ROUTINES FOR STARLET PATCH FACILITY
!
! History:
!       Author: Carol Peters, 21 Jul 1976: Version 01
!
!               Kathleen Morse, 19 Oct 1977: Version X01.00
!
! Modified by:
!
!       V03-002 MTR0016         Mike Rhodes             02-Nov-1982
!               Modify routine RELOCAT_INS to pass the address of the
!               the instruction(s) to be relocated to the patch area.
!               This address will be passed initially to PAT$EXP_AREA
!               which in turn may call routine PAT$BUILD_ISE (which is
!               called to create the default patch area if one does not
!               already exist).  PAT$BUILD_ISE will use this address to
!               propagate the image section attributes of the patched
!               image section to the newly created default patch area.
!
!       V03-001 MTR0012         Mike Rhodes             16-Aug-1982
!               Modify file names to remove duplicate file name useage
```

```
1 !                 between code and require files.
1 !
1 !         V02-023 PCG0001         Peter George            04-FEB-1981
1 !                 Add require statement for LIB$:PATDEF.REQ
1 !
1 !         V0122   BLS0039         Benn Schreiber          3-Feb-1981
1 !                 Correct handling of patch area.
1 !
1 !         V0121   CNH0014         Chris Hume              21-Sep-1979     11:00
1 !                 Added relocation support for the ACBG and ACBH instructions.
1 !
1 !         V0120   CNH0008         Chris Hume              28-Jun-1979     14:00
1 !                 Fix CASE replacement bug and disallow relocation of these
1 !                 instructions.  (PATMAI.B32 V0222, PATACT.B32 V0124,
1 !                 PATMAC.B32 V0217, PATMSG.MDL V0202)
1 !
1 ! Revision history:
1 !  NO   DATE            PROGRAMMER              PURPOSE
1 !  --   ----            ----------              -------
1 !
1 !  00   19-OCT-77       K.D. MORSE              ADAPT VERSION 49 FOR PATCH
1 !  01   01-DEC-77       K.D. MORSE              ADD DELETE ROUTINE.
1 !  02   27-DEC-77       K.D. MORSE              CHANGE PAT$OUT_VALUE CALLS. (57)
1 !                                               ADD PAT$SYM_DEPOS. (58)
1 !  03   2-JAN-78        K.D. MORSE              ADD CHECK FOR NO SYMBOLS IN IMAGE.
1 !  04   3-JAN-78        K.D. MORSE              ADD CHECK FOR NO PATCHAREA
1 !                                               ALLOCATED BEFORE DEPOSIT /PAT.
1 !  05   4-JAN-78        K.D. MORSE              NO CHANGES FOR VERS 50-53.
1 !  06   5-JAN-78        K.D. MORSE              CHANGE PAT$INS_DECODE CALLS. (54)
1 !                                               NO CHANGES FOR VERS 55,56.
1 !  07   24-JAN-78       K.D. MORSE              NO CHANGES FOR VERS 59.
1 !  08   27-JAN-78       K.D. MORSE              ADD CHECK FOR EXIT_TOKEN IN
1 !                                               PAT$REPLACE_CMD TO RECOGNIZE
1 !                                               END OF OLD LIST.
1 !  09   28-JAN-78       K.D. MORSE              BUILD REPLACEMENT CODE INTO
1 !                                               TEMPORARY BUFFER.
1 !  10   01-MAR-78       K.D. MORSE              CHANGE ERRONEOUS PAT$_DECODE
1 !                                               ERROR MSGS TO PAT$_ENCODE.
1 !  11   24-MAR-78       K.D. MORSE              NONE FOR VERS 60-61.
1 !  12   04-APR-78       K.D. MORSE              NONE FOR VERS 62.
1 !  13   25-APR-78       K.D. MORSE              CONVERT TO NATIVE COMPILER.
1 !  14   28-APR-78       K.D. MORSE              ADD ASSEMBLER DIRECTIVE FLAG
1 !                                               TO PAT$OUT_MEM_LOC.
1 !  15   18-MAY-78       K.D. MORSE              NO CHANGES FOR VERS 63.
1 !  16   26-MAY-78       K.D. MORSE              ADD CODE TO ALLOW FORWARD
1 !                                               REFERENCING IN SYMBOLIC
1 !                                               INSTRUCTION OPERANDS.
1 !  17   13-JUN-78       K.D. MORSE              ADD FAO COUNTS TO SIGNALS.
1 !  18   19-JUN-78       K.D. MORSE              NO CHANGES FOR VERS 64.
1 !  19   28-JUN-78       K.D. MORSE              NO CHANGES FOR VERS 65-67.
1 !                                               ADD CODE FOR EV/LITERAL AND
1 !                                               ROUTINE DISPLAY LVTS. (66)
1 !                                               NO CHANGES FOR VERS 69-74.
1 !
1 !--
```

```
;  114          0113  1 FORWARD ROUTINE
;  115          0114  1           PAT$DEPOSIT_CMD : NOVALUE,                       ! Deposits a datum into an address
;  116          0115  1           PAT$EXAMINE_CMD : NOVALUE,                       ! Examines a location
;  117          0116  1           PAT$REPLACE_CMD : NOVALUE,                       ! Replaces an instruction
;  118          0117  1           RELOCAT_INS : NOVALUE,                           ! Relocates instructions to patch area
;  119          0118  1           PAT$SUBST_INS,                                   ! Substitutes instructions in patch area
;  120          0119  1           PAT$OUT_MEM_LOC,                                 ! Outputs the contents of a memory location
;  121          0120  1           DISPLAY_LVTS : NOVALUE,                          ! Search LVT and display pathnames
;  122          0121  1           PAT$REG_MATCH,                                   ! Matches a string to a register name
;  123          0122  1           PAT$FILL_BUF : NOVALUE;                          ! Writes data into temporary buffers
;  124          0123  1
;  125          0124  1 LIBRARY 'SYS$LIBRARY:LIB.L32';                             ! System definitions
;  126          0125  1 REQUIRE 'SRC$:VXSMAC.REQ';
;  127          0190  1 REQUIRE 'SRC$:BSTRUC.REQ';
;  128          0266  1 REQUIRE 'SRC$:LISTEL.REQ';
;  129          0308  1 REQUIRE 'SRC$:PATPCT.REQ';
;  130          0348  1 REQUIRE 'SRC$:PATGEN.REQ';
;  131          0570  1 REQUIRE 'LIB$:PATDEF.REQ';                     ! Defines literals
;  132          0624  1 REQUIRE 'LIB$:PATMSG.REQ';
;  133          0798  1 REQUIRE 'SRC$:SYSLIT.REQ';
;  134          0848  1 REQUIRE 'SRC$:PATRTS.REQ';
;  135          1944  1 REQUIRE 'SRC$:SYSSER.REQ';
;  R1976  1      SWITCHES LIST (SOURCE);
;  R1977  1
;  R1978  1      EXTERNAL ROUTINE
;  R1979  1              PAT$fao_out;              ! formats a line and outputs to the terminal
;  R1980  1
```

PAT
V04

```
1 REQUIRE 'SRC$:PATTER.REQ';
1 REQUIRE 'SRC$:PREFIX.REQ';
1 REQUIRE 'SRC$:PATPRE.REQ';
1 REQUIRE 'SRC$:VAXOPS.REQ';
1
1 EXTERNAL
1               PAT$GB_SYMBOLS,                                 ! Indicator if image had symbols
1               PAT$GL_OLD_ASD,                                 ! Descriptor for old contents assembler dire
1               PAT$GL_NEW_ASD,                                 ! Descriptor for new contents assembler dire
1               PAT$GL_TEMP_BUF : BLOCK[,BYTE],                 ! Descriptor for temporary buffer for deposi
1               PAT$GL_RLOC_BUF : BLOCK[,BYTE],                 ! Descriptor for relocated instruction strea
1               PAT$GB_SUBST_IN : VECTOR[,BYTE],                ! Buffer for substitution instruction stream
1               PAT$GL_BR_DISPL,                                ! Branch displacement that does not fit
1               PAT$GL_PATAREA : REF BLOCK[,BYTE],              ! Pointer to patch area descriptor
1               PAT$GL_IMGHDR : REF BLOCK[,BYTE],               ! Pointer to image header
1               PAT$GB_LOC_TYPE: BYTE,                          ! Type of end range argument
1               PAT$GB_MOD_PTR: REF VECTOR [, BYTE],            ! Pointer to mode level
1               PAT$GL_IHPPTR : REF BLOCK[,BYTE],               ! Pointer to image header patch area
1               PAT$CP_OUT_STR : REF VECTOR[,BYTE],             ! Points into current output buffer
1               PAT$GL_CONTEXT: BITVECTOR,                      ! Context bits longword
1               PAT$GL_BUF_SIZ,                                 ! Holds count in output buffer
1               PAT$GL_HEAD_LST,                                ! Head of linked list of expressions
1               PAT$GL_LAST_LOC,                                ! Last location displayed
1               PAT$GL_LAST_VAL,                                ! Last value displayed
1               PAT$GL_NEXT_LOC,                                ! Next location to display
1               PAT$GL_SYMTBPTR,                                ! Pointer to current symbol table
1               PAT$GL_OLDLABLS,                                ! Listhead for old contents labels (from cur
1               PAT$GL_NEWLABLS,                                ! Listhead for new contents un-relocated lab
1               PAT$GL_RLCLABLS;                                ! List head for new contents relocated label
1
1 EXTERNAL ROUTINE
1               PAT$ADD_LABELS : NOVALUE,                       ! Adds labels to user-defined symbol table
1               PAT$ADD_NT_T_PV : NOVALUE,                      ! Build pathname vectors from NT_PTRs
1               PAT$EXP_AREA : NOVALUE,                         ! Expands patch area
1               PAT$FAO_PUT : NOVALUE,                          ! Formats buffered output
1               PAT$FREERELEASE,                                ! Deallocates free memory
1               PAT$FREEZ,                                      ! Allocates and zeroes free memory
1               PAT$GET_NXT_LVT,                                ! Provide access to the LVT
1               PAT$GET_VALUE : NOVALUE,                        ! Gets byte stream of values from image
1               PAT$INS_DECODE,                                 ! Routine to output memory as
1                                                               ! symbolic instructions
1               PAT$INS_ENCODE,                                 ! Routine to encode a symbolic instruction
1               PAT$MAP_ADDR : NOVALUE,                         ! Computes mapped addresses
1               PAT$OUT_NUM_VAL,                                ! Outputs numeric values
1               PAT$OUT_PUT : NOVALUE,                          ! Actually does the terminal I/O
1               PAT$OUT_SYM_VAL,                                ! Outputs symbol name with value
1               PAT$PRINT_PATH : NOVALUE,                       ! Print out pathnames
1               PAT$RESOLVE_INS : NOVALUE,                      ! Resolves forward references in symbolic in
1               PAT$SYMBOL_VALU,                                ! Finds the value bound to a symbol name
1               PAT$UNMAP_ADDR : NOVALUE,                       ! Computes unmapped addresses
1               PAT$VAL_TO_SYM,                                 ! Translates a value to a symbol name
1               PAT$WRITE_MEM;                                  ! Routine to write to user's memory
```

```
!++
!
! REGISTER_TABLE holds one entry per register.  Each entry is made
! up of one longword.  The first byte holds the character count of
! the register name.  The second through fourth bytes hold the
! register name string.  A sample entry follows:
!
!               +--------------------------------------------------------------+
!               !               !       0       !       R       !       2       !
!               +--------------------------------------------------------------+
!
!--

MACRO
        REGISTER_ENTRY (STRING) =
                %CHARCOUNT (STRING), %ASCII STRING, REP 3 - %CHARCOUNT (STRING) OF BYTE (0)%;

BIND
        REGISTER_TABLE = UPLIT BYTE (
                REGISTER_ENTRY ('R0'),
                REGISTER_ENTRY ('R1'),
                REGISTER_ENTRY ('R2'),
                REGISTER_ENTRY ('R3'),
                REGISTER_ENTRY ('R4'),
                REGISTER_ENTRY ('R5'),
                REGISTER_ENTRY ('R6'),
                REGISTER_ENTRY ('R7'),
                REGISTER_ENTRY ('R8'),
                REGISTER_ENTRY ('R9'),
                REGISTER_ENTRY ('R10'),
                REGISTER_ENTRY ('R11'),
                REGISTER_ENTRY ('AP'),
                REGISTER_ENTRY ('FP'),
                REGISTER_ENTRY ('SP'),
                REGISTER_ENTRY ('PC'),
                REGISTER_ENTRY ('PSL')):

        BLOCK [, LONG];

!++
! These field definitions control access to the register table.
!--
MACRO
        REG_NAME        =8, 24, 0%,
        CTD_REG_NAME    =0, 24, 0%,
        REG_CH_CNT      =0,  8, 0%;

!++
! Common ascii counted strings used in FAO calls.
!--

BIND
        CS_ASCII        = UPLIT ( %ASCIC '''!AD'''),
        COLON_TAB_STG   = UPLIT ( %ASCIC ':  '),
        CAR_CTL_STG     = UPLIT ( %ASCIC '!/'),
        OLD_TAB_STG     = UPLIT ( %ASCIC 'old:  '),
        NEW_TAB_STG     = UPLIT ( %ASCIC 'new:  ');
```

```
;  246          2903  1
;  247          2904  1 LITERAL
;  248          2905  1          NO_CASE_TABLE    = 0,                                         ! Don't print case dispatch table
;  249          2906  1          CASE_TABLE       = 1;                                         ! Print CASE dispatch table
```

```
1 SWITCHES NOOPTIMIZE;
1 GLOBAL ROUTINE PAT$DEPOSIT_CMD : NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       This routine handles all delete and deposit commands, those for
1 !       instructions and those for values.  The instruction(s)/value(s)
1 !       specified in the delete command must be identical to those actually
1 !       contained in the location(s), otherwise an error message is produced and
1 !       the command ends prematurely.  For a deposit command, the instruction(s)
1 !       /value(s) specified are written to consecutive locations.  The image is
1 !       not modified in either case unless the entire command works.
1 !
1 !       The command argument list is made up of entries that are
1 !       each three longwords long. The first is a forward link to the
1 !       next entry. The second longword in the first entry in the list
1 !       is the address into which some value(s) is (are) to be deleted or deposited.
1 !       The third longword is unused. The second longword in the second
1 !       and subsequent entries are the values to be deleted or deposited.
1 !       The first value should be the contents of the specified location;
1 !       the second, the contents of that location plus the current mode_length, etc.
1 !
1 !       For instructions, the increment is the length of each subsequent
1 !       instruction. The second longword contains the address of a counted byte
1 !       stream that is to be translated into a binary instruction which is the
1 !       old contents of that location.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       PAT$DEPOSIT_CMD ()
1 !
1 ! INPUTS:
1 !
1 !       none
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       PAT$GL_HEAD_LST, the head of the linked PATCH command argument list.
1 !       The current mode.
1 !
1 ! OUTPUTS:
1 !
1 !       none
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       The values PAT$GL_LAST_LOC, PAT$GL_LAST_VAL, and PAT$GL_NEXT_LOC
1 !       are set after each deposit is done.
1 !
1 ! ROUTINE VALUE:
1 !
1 !       novalue
1 !
1 ! SIDE EFFECTS:
1 !
1 !       The specified addresses have their values changed.
```

```
1 !       If a failure in a write occurs, the routine SIGNALs an error.
1 !
1 !--
1
2 BEGIN
2
2 LITERAL
2         NOP_INSTR       = 1,                             ! Fill char for instructions
2         ZERO_BYTE       = 0,                             ! Fill char for data
2         ONE_PAGE        = 1,                             ! Number of pages to expand patch area
2         MAX_INST_LEN    = 80;                            ! Maximum number of binary bytes in an instr
2
2 LOCAL
2         INSTRUC_BUF: VECTOR [MAX_INST_LEN, BYTE],
2         FILL_CHAR: BYTE,                                 ! Fill character for delete command
2         OLD_CONTENTS:   VECTOR[TTY_OUT_WIDTH, BYTE],     ! Buffer to hold old contents of location
2         MAPPED_LOC,                                      ! Mapped address of deposit destination
2         UNMAPPED_LOC,                                    ! Unmapped address of deposit destination
2         ISE_ADDR,                                        ! Image section entry for deposit destinatio
2         DEP_SRC_ADR,                                     ! Pointer to deposit source
2         DEP_SIZ,                                         ! Size of deposit to be made
2         POINTER;
2
2 !++
2 ! Check that all parameters were specified on the command.
2 !--
2 POINTER = .PAT$GL_HEAD_LST;
3 IF (.POINTER EQLA 0) OR (.LIST_ELEM_FLINK(.POINTER) EQLA 0)
2 THEN
2         SIGNAL (PAT$_INVCMD);
2
2 !++
2 ! Initialize unmapped address and PAT$GL_CONTEXT.  The context bit causes
2 ! the routine PAT$OUT_MEM_LOC to display a location instead of evaluating
2 ! a numeric expression.
2 !--
2 UNMAPPED_LOC = .LIST_ELEM_EXP1 (.POINTER);
2 PAT$GL_CONTEXT[EXAMINE_BIT] = TRUE;
2
2 !++
2 ! For DEPOSIT commands only:
2 !
2 ! Check if DEPOSIT qualifier, "/PATCH_AREA", was specified.
2 ! If so, check that the address specified is identical to the
2 ! start of the current patch area.  If it is not, report an
2 ! error and abort the DEPOSIT command.
2 !--
2 IF .PAT$GL_CONTEXT [PAT_AREA_BIT]
2 THEN
3         BEGIN
4         IF (.PAT$GL_PATAREA[DSC$W_LENGTH] EQL 0)
3         THEN
4                 IF (.PAT$GL_PATAREA [DSC$A_POINTER] EQLA .PAT$GL_IHPPTR[IHP$L_RW_PATADR])
3                 THEN
3                         PAT$EXP_AREA(ONE_PAGE);
4         IF (.PAT$GL_PATAREA [DSC$A_POINTER] NEQA .UNMAPPED_LOC)
3         THEN
```

```
3                       SIGNAL(PAT$_NOTPATADR, 2, .PAT$GL_PATAREA[DSC$A_POINTER], .UNMAPPED_LOC);
2               END;
2
2       !++
2       ! Set the fill character for DELETE commands.
2       !--
3       IF (.PAT$GB_MOD_PTR [MODE_INSTRUC])
2       THEN
3               BEGIN
3               PAT$GL_SYMTBPTR = .PAT$GL_NEWLABLS;                             ! Use new contents label table
3               FILL_CHAR = NOP_INSTR;                                          ! Fill character for instructions
3               END
2       ELSE
2               FILL_CHAR = ZERO_BYTE;                                          ! Fill character for data
2
2       !++
2       ! Loop to DEPOSIT (DELETE) all parameters specified in the command.
2       !--
2       REPEAT
3               BEGIN
3               POINTER = .LIST_ELEM_FLINK (.POINTER);
3
3               !++
3               ! Now determine the length of the instruction or data
3               ! which is to be deposited or deleted.
3               !--
3               IF .PAT$GB_MOD_PTR [MODE_INSTRUC]
3               THEN
4                       BEGIN
4                       !++
4                       ! This is a symbolic instruction to be deposited or deleted.
4                       ! It is currently in the form of a counted ASCII string that
4                       ! must be translated into binary form. The call to PAT$INS_ENCODE
4                       ! needs the address for which the instruction is encoded in
4                       ! order to resolve branches correctly.
4                       !--
4                       IF NOT PAT$INS_ENCODE (.LIST_ELEM_EXP1 (.POINTER),
4                               INSTRUC_BUF, .UNMAPPED_LOC,
5                               (IF .PAT$GL_CONTEXT[DELETE_BIT]
5                                       THEN PAT$GL_OLD_ASD
4                                       ELSE PAT$GL_NEW_ASD),
4                               PAT$GL_TEMP_BUF)
4                       THEN
4                               SIGNAL (PAT$_NOENCODE, 1, .LIST_ELEM_EXP1(.POINTER)); ! This instruction is invalid.
4                       DEP_SRC_ADR = INSTRUC_BUF [1];
4                       DEP_SIZ = .INSTRUC_BUF [0];
4                       END
3               ELSE
4                       BEGIN
4                       !++
4                       ! Determine length and address for deposits or deletes which are
4                       ! not symbolic instructions.  Then check for truncation of new value.
4                       !--
4                       DEP_SRC_ADR = LIST_ELEM_EXP1 (.POINTER);
4                       DEP_SIZ = .PAT$GB_MOD_PTR [MODE_LENGTH];
5                       IF (.LIST_ELEM_EXP1(.POINTER) LSS 0)
4                       THEN
```

```
5                               BEGIN
5                               IF .(LIST_ELEM_EXP1(.POINTER))<0, .DEP_SIZ*8, 1> NEQ .LIST_ELEM_EXP1(.POINTER)
5                               THEN
5                                       SIGNAL(PAT$_NUMTRUNC);
5                               END
4                       ELSE
4                               IF .(LIST_ELEM_EXP1(.POINTER))<0, .DEP_SIZ*8, 0> NEQ .LIST_ELEM_EXP1(.POINTER)
4                               THEN
4                                       SIGNAL(PAT$_NUMTRUNC);
3                       END;
3
3               !++
3               ! Now write the new values into a temporary buffer.  They are not
3               ! written directly into memory in case part of the command fails.
3               !--
3               PAT$FILL_BUF (PAT$GL_TEMP_BUF, .DEP_SRC_ADR, .DEP_SIZ);
3
3               !++
3               ! Finished with current value.  Reset last location,
3               ! next location, and last value, and exitloop.
3               !--
3               PAT$GL_LAST_LOC = .UNMAPPED_LOC;
3               UNMAPPED_LOC = .UNMAPPED_LOC + .DEP_SIZ;
3               IF NOT .PAT$GB_MOD_PTR [MODE_INSTRUC]
3               THEN
3                       PAT$GL_LAST_VAL = .LIST_ELEM_EXP1 (.POINTER);
3
3               !++
3               ! If there are no more values, then exit loop which builds
3               ! temporary deposit buffer.
3               !--
4               IF (.LIST_ELEM_FLINK (.POINTER) EQLA 0)
3               THEN
3                       EXITLOOP;
2               END;
2
2       !++
2       ! For DEPOSIT command only:
2       !
2       ! First check if this is writing into the patch area.  If so, check that there
2       ! is enough room in the patch area.  If not, then expand the patch area if
2       ! possible (that is, if the current patch area is the one defined int the image
2       ! header).  Otherwise, report an error and abort this command.
2       !--
2       IF .PAT$GL_CONTEXT[PAT_AREA_BIT]
2       THEN
3               BEGIN
4               IF (.PAT$GL_PATAREA[DSC$W_LENGTH] LSS .PAT$GL_TEMP_BUF[DSC$W_LENGTH])
3               THEN
4                       BEGIN
5                       IF (.PAT$GL_PATAREA[DSC$A_POINTER] EQLA .PAT$GL_IHPPTR[IHP$L_RW_PATADR])
4                       THEN
5                               PAT$EXP_AREA((.PAT$GL_TEMP_BUF[DSC$W_LENGTH] +
4                                               A_PAGE - 1)/A_PAGE)
4                       ELSE
4                               SIGNAL(PAT$_INSUFPAT, 2, .PAT$GL_TEMP_BUF[DSC$W_LENGTH],
4                                       .PAT$GL_PATAREA[DSC$A_POINTER],
4                                 .PAT$GL_PATAREA[DSC$W_LENGTH]);
3                       END;
2               END;
2
2       !++
2       ! Now resolve any forward references inside the symbolic instruction operands.
2       !--
2       PAT$RESOLVE_INS(PAT$GL_TEMP_BUF);
2
2       !++
2       ! Output the old values.
2       !--
2       PAT$GL_NEXT_LOC = .LIST_ELEM_EXP1(.PAT$GL_HEAD_LST);
2       WHILE .PAT$GL_NEXT_LOC LSSA .UNMAPPED_LOC
2       DO
2               PAT$OUT_MEM_LOC(.PAT$GL_NEXT_LOC, OLD_TAB_STG, PAT$GL_OLD_ASD, CASE_TABLE);
2
2       !++
2       ! For DELETE commands only:
2       !
2       ! Verify that the old values were actually in memory for DELETE commands.
2       ! Then fill the temporary buffer with the appropriate fill character.
2       !--
2       IF .PAT$GL_CONTEXT[DELETE_BIT]
2       THEN
3               BEGIN
3               !++
3               ! Now get the actual value in the location and
3               ! check that it equals the specified value.
3               !--
3               LOCAL
3                       BYTE_COUNT,                             ! Count of bytes verified
3                       BUF_SIZE;                               ! Size of old contents buffer to get
3               BYTE_COUNT = 0;
4               WHILE (.BYTE_COUNT LSS .PAT$GL_TEMP_BUF[DSC$W_LENGTH])
3               DO
4                       BEGIN
5                       IF ((BUF_SIZE = .PAT$GL_TEMP_BUF[DSC$W_LENGTH] - .BYTE_COUNT) GTR TTY_OUT_WIDTH)
4                       THEN
4                               BUF_SIZE = TTY_OUT_WIDTH;       ! Request only as much as buffer can hold
4                       PAT$GET_VALUE(.LIST_ELEM_EXP1(.PAT$GL_HEAD_LST)+.BYTE_COUNT,
4                                         .BUF_SIZE, OLD_CONTENTS);
4                       IF CH$NEQ(.BUF_SIZE, .PAT$GL_TEMP_BUF[DSC$A_POINTER]+.BYTE_COUNT,
4                                         .BUF_SIZE, OLD_CONTENTS)
4                       THEN
4                               SIGNAL(PAT$_DIFVAL+MSG$K_WARN);
4                       BYTE_COUNT = .BYTE_COUNT + .BUF_SIZE;
3                       END;
3               CH$FILL(.FILL_CHAR, .PAT$GL_TEMP_BUF[DSC$W_LENGTH],
3                         .PAT$GL_TEMP_BUF[DSC$A_POINTER]);
2               END;
2
2       !++
2       ! Now write the temporary buffer into memory.
2       !--
2       PAT$GL_NEXT_LOC = .LIST_ELEM_EXP1(.PAT$GL_HEAD_LST);
2       PAT$WRITE_MEM (.PAT$GL_NEXT_LOC, .PAT$GL_TEMP_BUF[DSC$A_POINTER],
2                       .PAT$GL_TEMP_BUF[DSC$W_LENGTH]);
2
2  !++
2  ! Output the new values.
2  !--
2  WHILE .PAT$GL_NEXT_LOC LSSA .UNMAPPED_LOC
2  DO
2          PAT$OUT_MEM_LOC(.PAT$GL_NEXT_LOC, NEW_TAB_STG, PAT$GL_NEW_ASD, CASE_TABLE);
2  !++
2  ! Now check if the deposit was into the current patch area.  If so,
2  ! update the patch area descriptor.
2  !--
2  IF .PAT$GL_CONTEXT [PAT_AREA_BIT]
2  THEN
3          BEGIN
3          PAT$GL_PATAREA[DSC$A_POINTER] = .PAT$GL_PATAREA[DSC$A_POINTER] +
3                                          .PAT$GL_TEMP_BUF[DSC$W_LENGTH];
3          PAT$GL_PATAREA[DSC$W_LENGTH] = .PAT$GL_PATAREA[DSC$W_LENGTH] -
3                                          .PAT$GL_TEMP_BUF[DSC$W_LENGTH];
3          END;
2  !++
2  ! Now add the new labels to the user-defined symbol table.
2  !--
2  PAT$ADD_LABELS(PAT$GL_NEWLABLS);
2
2
2  RETURN;
1  END;


                                          .TITLE  PATEXA
                                          .IDENT  \V04-000\

                                          .PSECT  _PAT$PLIT,NOWRT,NOEXE,0

       02  00000 P.AAA:   .BYTE   2
    30 52  00001          .ASCII  \R0\
       00  00003          .BYTE   0
       02  00004          .BYTE   2
    31 52  00005          .ASCII  \R1\
       00  00007          .BYTE   0
       02  00008          .BYTE   2
    32 52  00009          .ASCII  \R2\
       00  0000B          .BYTE   0
       02  0000C          .BYTE   2
    33 52  0000D          .ASCII  \R3\
       00  0000F          .BYTE   0
       02  00010          .BYTE   2
    34 52  00011          .ASCII  \R4\
       00  00013          .BYTE   0
       02  00014          .BYTE   2
    35 52  00015          .ASCII  \R5\
       00  00017          .BYTE   0
       02  00018          .BYTE   2
    36 52  00019          .ASCII  \R6\
       00  0001B          .BYTE   0
       02  0001C          .BYTE   2
```

```
                                    37  52  0001D            .ASCII  \R7\
                                        00  0001F            .BYTE   0
                                        02  00020            .BYTE   2
                                    38  52  00021            .ASCII  \R8\
                                        00  00023            .BYTE   0
                                        02  00024            .BYTE   2
                                    39  52  00025            .ASCII  \R9\
                                        00  00027            .BYTE   0
                                        03  00028            .BYTE   3
                                30  31  52  00029            .ASCII  \R10\
                                        03  0002C            .BYTE   3
                                31  31  52  0002D            .ASCII  \R11\
                                        02  00030            .BYTE   2
                                    50  41  00031            .ASCII  \AP\
                                        00  00033            .BYTE   0
                                        02  00034            .BYTE   2
                                    50  46  00035            .ASCII  \FP\
                                        00  00037            .BYTE   0
                                        02  00038            .BYTE   2
                                    50  53  00039            .ASCII  \SP\
                                        00  0003B            .BYTE   0
                                        02  0003C            .BYTE   2
                                    43  50  0003D            .ASCII  \PC\
                                        00  0003F            .BYTE   0
                                        03  00040            .BYTE   3
                            4C  53  50  00041            .ASCII  \PSL\
    00  00  27  44  41  21  27  05  00044 P.AAB:     .ASCII  <5>\'!AD'\<0><0>
                        20  20  3A  03  0004C P.AAC:     .ASCII  <3>\:  \
                        00  2F  21  02  00050 P.AAD:     .ASCII  <2>\!/\<0>
    00  00  09  3A  64  6C  6F  05  00054 P.AAE:     .ASCII  <5>\old:\<9><0><0>
    00  00  09  3A  77  65  6E  05  0005C P.AAF:     .ASCII  <5>\new:\<9><0><0>

                                              ISE$C_SIZE==        20
                                              TXT$C_SIZE==        4
                                              PAL$C_SIZE==        16
                                              ASD$C_SIZE==        9
                                              FWR$C_SIZE==        24
                                              REGISTER_TABLE=     P.AAA
                                              CS_ASCII=           P.AAB
                                              COLON_TAB_STG=      P.AAC
                                              CAR_CTL_STG=        P.AAD
                                              OLD_TAB_STG=        P.AAE
                                              NEW_TAB_STG=        P.AAF
                                                         .EXTRN  PAT$FAO_OUT, PAT$GB_SYMBOLS
                                                         .EXTRN  PAT$GL_OLD_ASD, PAT$GL_NEW_ASD
                                                         .EXTRN  PAT$GL_TEMP_BUF
                                                         .EXTRN  PAT$GL_RLOC_BUF
                                                         .EXTRN  PAT$GB_SUBST_IN
                                                         .EXTRN  PAT$GL_BR_DISPL
                                                         .EXTRN  PAT$GL_PATAREA, PAT$GL_IMGHDR
                                                         .EXTRN  PAT$GB_LOC_TYPE
                                                         .EXTRN  PAT$GB_MOD_PTR, PAT$GL_IHPPTR
                                                         .EXTRN  PAT$CP_OUT_STR, PAT$GL_CONTEXT
                                                         .EXTRN  PAT$GL_BUF_SIZ, PAT$GL_HEAD_LST
                                                         .EXTRN  PAT$GL_LAST_LOC
                                                         .EXTRN  PAT$GL_LAST_VAL
                                                         .EXTRN  PAT$GL_NEXT_LOC
                                              .EXTRN  PAT$GL_SYMTBPTR
                                              .EXTRN  PAT$GL_OLDLABLS
                                              .EXTRN  PAT$GL_NEWLABLS
                                              .EXTRN  PAT$GL_RLCLABLS
                                              .EXTRN  PAT$ADD_LABELS, PAT$ADD_NT_T_PV
                                              .EXTRN  PAT$EXP_AREA, PAT$FAO_PUT
                                              .EXTRN  PAT$FREERELEASE
                                              .EXTRN  PAT$FREEZ, PAT$GET_NXT_LVT
                                              .EXTRN  PAT$GET_VALUE, PAT$INS_DECODE
                                              .EXTRN  PAT$INS_ENCODE, PAT$MAP_ADDR
                                              .EXTRN  PAT$OUT_NUM_VAL
                                              .EXTRN  PAT$OUT_PUT, PAT$OUT_SYM_VAL
                                              .EXTRN  PAT$PRINT_PATH, PAT$RESOLVE_INS
                                              .EXTRN  PAT$SYMBOL_VALU
                                              .EXTRN  PAT$UNMAP_ADDR, PAT$VAL_TO_SYM
                                              .EXTRN  PAT$WRITE_MEM
                                              .WEAK   ACCESS_CHECK

                                              .PSECT  _PAT$CODE,NOWRT,2

                                 0FFC 00000   .ENTRY  PAT$DEPOSIT_CMD, Save R2,R3,R4,R5,R6,R7,R8,-; 2908
                                                      R9,R10,R11                                  ;
             5B 00000000G  00  9E 00002       MOVAB   LIB$SIGNAL, R11                             ;
             5A 00000000G  EF  9E 00009       MOVAB   PAT$GL_NEXT_LOC, R10                        ;
             59 00000000G  EF  9E 00010       MOVAB   PAT$GL_PATAREA, R9                          ;
             58 00000000G  EF  9E 00017       MOVAB   PAT$GL_TEMP_BUF, R8                         ;
             5E      FF2C  CE  9E 0001E       MOVAB   -212(SP), SP                                ;
             52 00000000G  EF  D0 00023       MOVL    PAT$GL_HEAD_LST, POINTER                    ; 2990
                           04  13 0002A       BEQL    1$                                          ; 2991
                           62  D5 0002C       TSTL    (POINTER)                                   ;
                           09  12 0002E       BNEQ    2$                                          ;
                006D80DA   8F  DD 00030 1$:   PUSHL   #7176410                                    ; 2993
             6B            01  FB 00036       CALLS   #1, LIB$SIGNAL                              ;
             56        04  A2  D0 00039 2$:   MOVL    4(POINTER), UNMAPPED_LOC                    ; 3000
   00000000G EF            01  88 0003D       BISB2   #1, PAT$GL_CONTEXT+1                        ; 3001
3B 00000000G EF            03  E1 00044       BBC     #3, PAT$GL_CONTEXT+2, 4$                    ; 3011
                       00  B9  B5 0004C       TSTW    @PAT$GL_PATAREA                             ; 3014
                           1A  12 0004F       BNEQ    3$                                          ;
             51            69  D0 00051       MOVL    PAT$GL_PATAREA, R1                          ; 3016
             50 00000000G  EF  D0 00054       MOVL    PAT$GL_IHPPTR, R0                           ;
          14 A0        04  A1  D1 0005B       CMPL    4(R1), 20(R0)                               ;
                           09  12 00060       BNEQ    3$                                          ;
                           01  DD 00062       PUSHL   #1                                          ; 3018
   00000000G EF            01  FB 00064       CALLS   #1, PAT$EXP_AREA                            ;
             50            69  D0 0006B 3$:   MOVL    PAT$GL_PATAREA, R0                          ; 3019
             56        04  A0  D1 0006E       CMPL    4(R0), UNMAPPED_LOC                         ;
                           13  13 00072       BEQL    4$                                          ;
                           56  DD 00074       PUSHL   UNMAPPED_LOC                                ; 3021
             50            69  D0 00076       MOVL    PAT$GL_PATAREA, R0                          ;
                       04  A0  DD 00079       PUSHL   4(R0)                                       ;
                           02  DD 0007C       PUSHL   #2                                          ;
                006D8122   8F  DD 0007E       PUSHL   #7176482                                    ;
             6B            04  FB 00084       CALLS   #4, LIB$SIGNAL                              ;
             50 00000000G  EF  D0 00087 4$:   MOVL    PAT$GB_MOD_PTR, R0                          ; 3027
             10        03  A0  E9 0008E       BLBC    3(R0), 5$                                   ;
   00000000G EF 00000000G  EF  D0 00092       MOVL    PAT$GL_NEWLABLS, PAT$GL_SYMTBPTR            ; 3030
             57            01  90 0009D       MOVB    #1, FILL_CHAR                               ; 3031
```

```
PATEXA                                                   B 5
V04-000                                                  16-Sep-1984 00:30:29    VAX-11 Bliss-32 V4.0-742                   Page  16
                                                         14-Sep-1984 12:52:32    DISK$VMSMASTER:[PATCH.SRC]PATEXA.B32;1      (4)

                                            02  11 000A0        BRB     6$                                        : 3027
                                            57  94 000A2 5$:    CLRB    FILL_CHAR                                 : 3034
                          52                62  D0 000A4 6$:    MOVL    (POINTER), POINTER                        : 3041
                          50 00000000G      EF  D0 000A7        MOVL    PAT$GB_MOD_PTR, R0                        : 3047
                          46         03     A0  E9 000AE        BLBC    3(R0), 10$                                :
                                            58  DD 000B2        PUSHL   R8                                        : 3057
              09 00000000G EF               06  E1 000B4        BBC     #6, PAT$GL_CONTEXT+2, 7$                  : 3059
                          50 00000000G      EF  9E 000BC        MOVAB   PAT$GL_OLD_ASD, R0                        :
                                            07  11 000C3        BRB     8$                                        :
                          50 00000000G      EF  9E 000C5 7$:    MOVAB   PAT$GL_NEW_ASD, R0                        :
                                            50  DD 000CC 8$:    PUSHL   R0                                        :
                                            56  DD 000CE        PUSHL   UNMAPPED_LOC                              : 3058
                                     B0     AD  9F 000D0        PUSHAB  INSTRUC_BUF                               : 3057
                                     04     A2  DD 000D3        PUSHL   4(POINTER)                                :
                 00000000G EF               05  FB 000D6        CALLS   #5, PAT$INS_ENCODE                        :
                           0E               50  E8 000DD        BLBS    R0, 9$                                    :
                                     04     A2  DD 000E0        PUSHL   4(POINTER)                                : 3064
                                            01  DD 000E3        PUSHL   #1                                        :
                             006D810A       8F  DD 000E5        PUSHL   #7176458                                  :
                          6B                03  FB 000EB        CALLS   #3, LIB$SIGNAL                            :
                          54         B1     AD  9E 000EE 9$:    MOVAB   INSTRUC_BUF+1, DEP_SRC_ADR                : 3065
                          53         B0     AD  9A 000F2        MOVZBL  INSTRUC_BUF, DEP_SIZ                      : 3066
                                            39  11 000F6        BRB     13$                                       : 3047
                          54         04     A2  9E 000F8 10$:   MOVAB   4(R2), DEP_SRC_ADR                        : 3074
                          50 00000000G      EF  D0 000FC        MOVL    PAT$GB_MOD_PTR, R0                        : 3075
                          53         01     A0  9A 00103        MOVZBL  1(R0), DEP_SIZ                            :
                                     04     A2  D5 00107        TSTL    4(POINTER)                                : 3076
                                            0C  18 0010A        BGEQ    11$                                       :
              50          53                03  78 0010C        ASHL    #3, DEP_SIZ, R0                           : 3079
 51     04    A2          50                00  EE 00110        EXTV    #0, R0, 4(POINTER), R1                    :
                                            0A  11 00116        BRB     12$                                       :
              50          53                03  78 00118 11$:   ASHL    #3, DEP_SIZ, R0                           : 3084
 51     04    A2          50                00  EF 0011C        EXTZV   #0, R0, 4(POINTER), R1                    :
                     04   A2                51  D1 00122 12$:   CMPL    R1, 4(POINTER)                            :
                                            09  13 00126        BEQL    13$                                       :
                             006D8023       8F  DD 00128        PUSHL   #7176227                                  : 3086
                          6B                01  FB 0012E        CALLS   #1, LIB$SIGNAL                            :
                                            53  DD 00131 13$:   PUSHL   DEP_SIZ                                   : 3093
                                            54  DD 00133        PUSHL   DEP_SRC_ADR                               :
                                            58  DD 00135        PUSHL   R8                                        :
                 00000000V EF               03  FB 00137        CALLS   #3, PAT$FILL_BUF                          :
                 00000000G EF               56  D0 0013E        MOVL    UNMAPPED_LOC, PAT$GL_LAST_LOC             : 3099
                           56               53  C0 00145        ADDL2   DEP_SIZ, UNMAPPED_LOC                     : 3100
                          50 00000000G      EF  D0 00148        MOVL    PAT$GB_MOD_PTR, R0                        : 3101
                          08         03     A0  E8 0014F        BLBS    3(R0), 14$                                :
                 00000000G EF        04     A2  D0 00153        MOVL    4(POINTER), PAT$GL_LAST_VAL               : 3103
                                            62  D5 0015B 14$:   TSTL    (POINTER)                                 : 3109
                                            03  13 0015D        BEQL    15$                                       :
                                          FF42  31 0015F        BRW     6$                                        :
              48 00000000G EF               03  E1 00162 15$:   BBC     #3, PAT$GL_CONTEXT+2, 17$                 : 3122
                          68         00     B9  B1 0016A        CMPW    @PAT$GL_PATAREA, PAT$GL_TEMP_BUF          : 3125
                                            42  1E 0016E        BGEQU   17$                                       :
                          51                69  D0 00170        MOVL    PAT$GL_PATAREA, R1                        : 3128
                          50 00000000G      EF  D0 00173        MOVL    PAT$GL_IHPPTR, R0                         :
                     14   A0         04     A1  D1 0017A        CMPL    4(R1), 20(R0)                             :
                                            19  12 0017F        BNEQ    16$                                       :
                          50                68  3C 00181        MOVZWL  PAT$GL_TEMP_BUF, R0                       : 3130
                          50      01FF   C0  9E 00184          MOVAB   511(R0), R0                                     ;
                  7E      50 00000200   8F  C7 00189          DIVL3   #512, R0, -(SP)                                 ; 3131
               00000000G  EF            01  FB 00191          CALLS   #1, PAT$EXP_AREA                                ;
                          18            11 00198              BRB     17$                                             ; 3130
                          7E        00   B9  3C 0019A 16$:    MOVZWL  @PAT$GL_PATAREA, -(SP)                          ; 3135
                          50            69  D0 0019E          MOVL    PAT$GL_PATAREA, R0                              ; 3134
                                    04   A0  DD 001A1          PUSHL   4(R0)                                           ;
                          7E            68  3C 001A4          MOVZWL  PAT$GL_TEMP_BUF, -(SP)                          ; 3133
                                        02  DD 001A7          PUSHL   #2                                              ;
                             006D80C2   8F  DD 001A9          PUSHL   #7176386                                        ;
                          6B            05  FB 001AF          CALLS   #5, LIB$SIGNAL                                  ;
                                        58  DD 001B2 17$:     PUSHL   R8                                              ; 3142
               00000000G  EF            01  FB 001B4          CALLS   #1, PAT$RESOLVE_INS                             ;
                          50 00000000G  EF  D0 001BB          MOVL    PAT$GL_HEAD_LST, R0                             ; 3147
                          6A        04  A0  D0 001C2          MOVL    4(R0), PAT$GL_NEXT_LOC                          ;
                          56            6A  D1 001C6 18$:     CMPL    PAT$GL_NEXT_LOC, UNMAPPED_LOC                   ; 3148
                                        19  1E 001C9          BGEQU   19$                                             ;
                                        01  DD 001CB          PUSHL   #1                                              ; 3150
                             00000000G  EF  9F 001CD          PUSHAB  PAT$GL_OLD_ASD                                  ;
                             00000000'  EF  9F 001D3          PUSHAB  OLD_TAB_STG                                     ;
                                        6A  DD 001D9          PUSHL   PAT$GL_NEXT_LOC                                 ;
               00000000V  EF            04  FB 001DB          CALLS   #4, PAT$OUT_MEM_LOC                             ;
                                        E2  11 001E2          BRB     18$                                             ;
            4F 00000000G  EF            06  E1 001E4 19$:     BBC     #6, PAT$GL_CONTEXT+2, 24$                       ; 3158
                                        54  D4 001EC          CLRL    BYTE_COUNT                                      ; 3168
       54         68                10  00  ED 001EE 20$:     CMPZV   #0, #16, PAT$GL_TEMP_BUF, BYTE_COUNT            ; 3169
                                        3F  15 001F3          BLEQ    23$                                             ;
                          55            68  3C 001F5          MOVZWL  PAT$GL_TEMP_BUF, BUF_SIZE                       ; 3172
                          55            54  C2 001F8          SUBL2   BYTE_COUNT, BUF_SIZE                            ;
               00000084   8F            55  D1 001FB          CMPL    BUF_SIZE, #132                                  ;
                                        04  15 00202          BLEQ    21$                                             ;
                          55        84   8F  9A 00204          MOVZBL  #132, BUF_SIZE                                  ; 3174
                                  4020   8F  BB 00208 21$:     PUSHR   #^M<R5,SP>                                      ; 3176
                          50 00000000G  EF  D0 0020C          MOVL    PAT$GL_HEAD_LST, R0                             ; 3175
                                  04 B044   9F 00213          PUSHAB  @4(R0)[BYTE_COUNT]                              ;
               00000000G  EF            03  FB 00217          CALLS   #3, PAT$GET_VALUE                               ;
                  6E      04 B844       55  29 0021E          CMPC3   BUF_SIZE, @PAT$GL_TEMP_BUF+4[BYTE_COUNT], -     ; 3177
                                                                      OLD_CONTENTS                                    ;
                                        09  13 00224          BEQL    22$                                             ;
                             006D8290   8F  DD 00226          PUSHL   #7176848                                        ; 3180
                          6B            01  FB 0022C          CALLS   #1, LIB$SIGNAL                                  ;
                          54            55  C0 0022F 22$:     ADDL2   BUF_SIZE, BYTE_COUNT                            ; 3181
                                        BA  11 00232          BRB     20$                                             ; 3169
       68         57            6E      00  2C 00234 23$:     MOVC5   #0, (SP), FILL_CHAR, PAT$GL_TEMP_BUF, -         ; 3184
                                    04  B8    00239                   @PAT$GL_TEMP_BUF+4                              ;
                          50 00000000G  EF  D0 0023B 24$:     MOVL    PAT$GL_HEAD_LST, R0                             ; 3190
                          6A        04  A0  D0 00242          MOVL    4(R0), PAT$GL_NEXT_LOC                          ;
                          7E            68  3C 00246          MOVZWL  PAT$GL_TEMP_BUF, -(SP)                          ; 3192
                                    04  A8  DD 00249          PUSHL   PAT$GL_TEMP_BUF+4                               ; 3191
                                        6A  DD 0024C          PUSHL   PAT$GL_NEXT_LOC                                 ;
               00000000G  EF            03  FB 0024E          CALLS   #3, PAT$WRITE_MEM                               ;
                          56            6A  D1 00255 25$:     CMPL    PAT$GL_NEXT_LOC, UNMAPPED_LOC                   ; 3197
                                        19  1E 00258          BGEQU   26$                                             ;
                                        01  DD 0025A          PUSHL   #1                                              ; 3199
                             00000000G  EF  9F 0025C          PUSHAB  PAT$GL_NEW_ASD                                  ;
                             00000000'  EF  9F 00262          PUSHAB  NEW_TAB_STG                                     ;
                                        6A  DD 00268          PUSHL   PAT$GL_NEXT_LOC                                 ;
```

```
                        0000000V  EF            04  FB 0026A         CALLS    #4, PAT$OUT_MEM_LOC                                   ;
                                                E2  11 00271         BRB      25$                                                   ;
                     13 00000000G EF            03  E1 00273 26$:    BBC      #3, PAT$GL_CONTEXT+2, 27$                             ; 3204
                                  50            69  D0 0027B         MOVL     PAT$GL_PATAREA, R0                                    ; 3207
                                  51            69  D0 0027E         MOVL     PAT$GL_PATAREA, R1                                    ;
                                  52            68  3C 00281         MOVZWL   PAT$GL_TEMP_BUF, R2                                   ; 3208
                             04   A0      04 B142  9E 00284         MOVAB    @4(R1)[R2], 4(R0)                                     ;
                             00   B9            68  A2 0028A         SUBW2    PAT$GL_TEMP_BUF, @PAT$GL_PATAREA                      ; 3210
                                      00000000G EF  9F 0028E 27$:    PUSHAB   PAT$GL_NEWLABLS                                       ; 3215
                        00000000G EF            01  FB 00294         CALLS    #1, PAT$ADD_LABELS                                    ;
                                                    04 0029B         RET                                                            ; 3219

; Routine Size:  668 bytes,    Routine Base:  _PAT$CODE + 0000


;  564          3220  1 SWITCHES OPTIMIZE;
```

```
1 GLOBAL ROUTINE PAT$EXAMINE_CMD : NOVALUE =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Examines a list of addresses.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       PAT$EXAMINE_CMD ()
1 !
1 ! INPUTS:
1 !
1 !       none
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       The address of the first element of a list of addresses.
1 !       The last address examined, and the next logical address to examine.
1 !
1 ! OUTPUTS:
1 !
1 !       none
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       New values for last and next location, and last value
1 !
1 ! ROUTINE VALUE:
1 !
1 !       novalue
1 !
1 ! SIDE EFFECTS:
1 !
1 !       The values of various addresses are output.
1 !       If an error occurs, the routine returns without further
1 !       processing except to output an error message to the output
1 !       device.
1 !
1 !--
1
2 BEGIN
2
2 LOCAL
2       MAPPED_NEXT_LOC,                                ! Mapped address of next location
2       ISE_ADDR,                                       ! ISE address for mapped address
2       POINTER;
2
2 POINTER = .PAT$GL_HEAD_LST;
3 IF (.POINTER EQL 0)
2 THEN
2
2       !++
2       ! No location was specified. Examine the next location in sequence.
2       !--
2       PAT$OUT_MEM_LOC (.PAT$GL_NEXT_LOC, 0, PAT$GL_OLD_ASD, CASE_TABLE)
2 ELSE DO
```

```
;  623          3278  3          BEGIN
;  624          3279  3
;  625          3280  3          LOCAL
;  626          3281  3                  LAST_LOC;
;  627          3282  3
;  628          3283  3          !++
;  629          3284  3          ! Pick up the next value which we will try to
;  630          3285  3          ! display and copy it into LAST_LOC.
;  631          3286  3          !--
;  632          3287  3          LAST_LOC = .LIST_ELEM_EXP1 (.POINTER);
;  633          3288  3
;  634          3289  3          !++
;  635          3290  3          ! If the end range argument is null, then make it the same as the start
;  636          3291  3          ! range argument so that only one location will be displayed.
;  637          3292  3          !--
;  638          3293  3          IF .LIST_ELEM_EXP2 (.POINTER) EQL 0
;  639          3294  3          THEN LIST_ELEM_EXP2 (.POINTER) = .LIST_ELEM_EXP1 (.POINTER);
;  640          3295  3
;  641          3296  3          !++
;  642          3297  3          ! Check for range reversal.
;  643          3298  3          !--
;  644          3299  4          IF( .LIST_ELEM_EXP2(.POINTER) LSSA .LIST_ELEM_EXP1(.POINTER) )
;  645          3300  3          THEN
;  646          3301  4                  BEGIN
;  647          3302  4                  SIGNAL (PAT$_EXARANGE);
;  648          3303  4                  RETURN;
;  649          3304  3                  END;
;  650          3305  3          WHILE (.LAST_LOC LEQ .LIST_ELEM_EXP2 (.POINTER)) DO
;  651          3306  4                  BEGIN
;  652          3307  4                  IF NOT PAT$OUT_MEM_LOC (.LAST_LOC, 0, PAT$GL_OLD_ASD, CASE_TABLE)
;  653          3308  4                  THEN RETURN;
;  654          3309  4                  LAST_LOC = .PAT$GL_NEXT_LOC;
;  655          3310  3                  END;
;  656          3311  3          END
;  657          3312  2  UNTIL (POINTER = .LIST_ELEM_FLINK (.POINTER)) EQL 0;
;  658          3313  2
;  659          3314  1  END;



                                    007C 00000          .ENTRY  PAT$EXAMINE_CMD, Save R2,R3,R4,R5,R6          ; 3221
             56 00000000V   EF   9E 00002          MOVAB   PAT$OUT_MEM_LOC, R6                                  ;
             55 00000000G   EF   9E 00009          MOVAB   PAT$GL_NEXT_LOC, R5                                  ;
             54 00000000G   EF   9E 00010          MOVAB   PAT$GL_OLD_ASD, R4                                   ;
             52 00000000G   EF   D0 00017          MOVL    PAT$GL_HEAD_LST, POINTER                             ; 3269
                            0C   12 0001E          BNEQ    1$                                                   ; 3270
                            01   DD 00020          PUSHL   #1                                                   ; 3276
                            54   DD 00022          PUSHL   R4                                                   ;
                            7E   D4 00024          CLRL    -(SP)                                                ;
                            65   DD 00026          PUSHL   PAT$GL_NEXT_LOC                                      ;
             66             04   FB 00028          CALLS   #4, PAT$OUT_MEM_LOC                                  ;
                                 04 0002B          RET                                                          ;
             53        04   A2   D0 0002C 1$:      MOVL    4(POINTER), LAST_LOC                                 ; 3287
                       08   A2   D5 00030          TSTL    8(POINTER)                                           ; 3293
                            05   12 00033          BNEQ    2$                                                   ;
```

```
                08   A2          04   A2  D0 00035        MOVL    4(POINTER), 8(POINTER)                              ; 3294
                04   A2          08   A2  D1 0003A 2$:    CMPL    8(POINTER), 4(POINTER)                              ; 3299
                                          0E  1E 0003F        BGEQU   3$
                            006D80AA   8F  DD 00041        PUSHL   #7176362                                            ; 3302
          00000000G   00               01  FB 00047        CALLS   #1, LIB$SIGNAL
                                          04 0004E        RET                                                         ; 3301
                08   A2               53  D1 0004F 3$:    CMPL    LAST_LOC, 8(POINTER)                                ; 3305
                                       13  14 00053        BGTR    4$
                                       01  DD 00055        PUSHL   #1                                                  ; 3307
                                       54  DD 00057        PUSHL   R4                                                  ;
                                       7E  D4 00059        CLRL    -(SP)                                               ;
                                       53  DD 0005B        PUSHL   LAST_LOC                                            ;
                     66                04  FB 0005D        CALLS   #4, PAT$OUT_MEM_LOC                                 ;
                     0A                50  E9 00060        BLBC    R0, 5$                                              ;
                     53                65  D0 00063        MOVL    PAT$GL_NEXT_LOC, LAST_LOC                           ; 3309
                                       E7  11 00066        BRB     3$                                                  ; 3305
                     52                62  D0 00068 4$:    MOVL    (POINTER), POINTER                                  ; 3312
                                       BF  12 0006B        BNEQ    1$
                                          04 0006D 5$:    RET                                                         ; 3314

; Routine Size:  110 bytes,    Routine Base:  _PAT$CODE + 029C
```

```
GLOBAL ROUTINE PAT$REPLACE_CMD : NOVALUE =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This routine handles all REPLACE, INSERT and VERIFY commands, those for
!       instructions and those for values.  The instruction/value specified in
!       the command must be identical to those actually contained in the
!       location, otherwise an error message is produced and the command
!       ends prematurely.
!
!       The command argument list is made up of entries that are
!       each three longwords long. The first is a forward link to the
!       next entry. The second longword in the first entry in the list
!       is the address into which some value(s) is (are) to be replaced.
!       The third longword is unused. The second longword in successive
!       arguments is the old values/instructions and thier replacemeents.
!       The first value should be the contents of the specified location;
!       the second, the contents of that location plus the current mode_length,
!       etc.  The last old value has an EXIT_TOKEN in the third longword
!       (all others have zero).
!
!       For instruction replacement, the second longword contains
!       the address of a counted ascii stream that is to be translated
!       into a binary instruction.
!
!       The VERIFY command is identical to the replace command save
!       that it has no replacement values.  The INSERT command has only one
!       old instruction and causes it to be moved to the patch area instead
!       of replaced.
!
!       There are three label tables used to differentiate between old labels,
!       new un-relocated labels, and new relocated labels.  The table(s) used
!       to resolve symbols inside symbolic instructions depends updon which
!       instruction is being encoded (old or new) and whether or not they are
!       being relocated to patch area.  Three tables are necessary to handle
!       relocation correctly, i.e., old labels can be used for all instructions
!       but new labels cannot be used for relocated instructions.  No labels
!       can be added to the user-defined symbol table until the PATCH command
!       is successfully executed.
!
! CALLING SEQUENCE:
!
!       PAT$REPLACE_CMD ()
!
! INPUTS:
!
!       none
!
! IMPLICIT INPUTS:
!
!       PAT$GL_HEAD_LST, the head of the linked PATCH command argument list.
!       the current mode, and the current patch area descriptor.
!
! OUTPUTS:
!
!       none
```

```
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       none
1 !
1 ! ROUTINE VALUE:
1 !
1 !       novalue
1 !
1 ! SIDE EFFECTS:
1 !
1 !       If a failure in a write or contents verification occurs,
1 !       the routine returns immediately.  If the command is executed
1 !       successfully, then the specified addresses have new values and any
1 !       labels in the command are added to the user-defined symbol table.
1 !
1 !--
1
2 BEGIN
2
2 LITERAL
2       NOP_INSTR = 1,                                          ! Fill character for instructions
2       ZERO_BYTE = 0,                                          ! Fill characters for data
2       MAX_INST_LEN    = 80;                                   ! Maximum number of binary bytes in an instr
2
2 LOCAL
2       BYTE_COUNT,                                             ! Count of bytesverified
2       BUF_SIZE,                                               ! Size of OLD_CONTENTS to compare
2       INSTRUC_BUF: VECTOR [MAX_INST_LEN, BYTE],
2       OLD_CONTENTS : VECTOR[TTY_OUT_WIDTH,BYTE],              ! Buffer to hold actual contents of location
2       UNMAPPED_LOC,                                           ! Unmapped address of DEPOSIT destination
2       OLD_VALUE_PTR : REF VECTOR[,BYTE],                      ! Pointer of actual contents of location
2       VAL_SIZ,                                                ! Size of current value/instruction
2       HOLE_SIZ,                                               ! Cumulative size to replace
2       NEXT_LOC,                                               ! Pointer to next consecutive location
2       FILL_CHAR : BYTE,                                       ! Fill character for data/instructions
2       POINTER,
2       OLD_INS_SIZ,                                            ! Number of bytes of old instruction moved f
2       NEW_INS_PTR;                                            ! Pointer to first new instruction argument
2
2 !++
2 ! Check for required parameter.
2 !--
2 POINTER = .PAT$GL_HEAD_LST;
3 IF  (.POINTER EQLA 0) OR (.LIST_ELEM_FLINK (.POINTER) EQLA 0)
2 THEN
2       SIGNAL(PAT$_INVCMD);
2
2 !++
2 ! Set the examine bit for PAT$OUT_MEM_LOC.
2 !
2 PAT$GL_CONTEXT [EXAMINE_BIT] = TRUE;
2 UNMAPPED_LOC = .LIST_ELEM_EXP1 (.POINTER);
2 NEXT_LOC = .LIST_ELEM_EXP1 (.POINTER);
2 HOLE_SIZ = 0;
2 PAT$GL_SYMTBPTR = .PAT$GL_OLDLABLS;                            ! Use old contents label list
2
```

```
!++
! Now loop, validating the old contents are the expected values.
! The last old value argument contains an EXIT_TOKEN in position LIST_ELEM_EXP2.
!--
REPEAT
        BEGIN
        POINTER = .LIST_ELEM_FLINK (.POINTER);

        !++
        ! Now compute the binary instruction stream that should be in
        ! the location.
        !--
        IF .PAT$GB_MOD_PTR [MODE_INSTRUC]
        THEN
                BEGIN
                !++
                ! This is a symbolic instruction.  It is currently
                ! in the form of a counted ASCII string that must be translated
                ! into binary form. The call to PAT$INS_ENCODE needs the address
                ! into which the instruction is being deposited in order to
                ! resolve branches correctly.
                !--
                IF NOT PAT$INS_ENCODE (.LIST_ELEM_EXP1 (.POINTER),
                        INSTRUC_BUF, .NEXT_LOC, PAT$GL_OLD_ASD, PAT$GL_TEMP_BUF)
                THEN
                        SIGNAL (PAT$_NOENCODE, 1, .LIST_ELEM_EXP1(.POINTER)); ! This instruction is not vali
                OLD_VALUE_PTR = INSTRUC_BUF [1];
                VAL_SIZ = .INSTRUC_BUF [0];
                END
        ELSE
                BEGIN
                !++
                ! Value is not an instruction.  Therefore it is on the parse
                ! stack.  The current mode for length tells the number of bytes
                ! of the value.  Set pointer to data and size indicator.  Then
                ! check for a truncation error.
                !--
                OLD_VALUE_PTR = LIST_ELEM_EXP1 (.POINTER);
                VAL_SIZ = .PAT$GB_MOD_PTR [MODE_LENGTH];
                IF .LIST_ELEM_EXP1(.POINTER) LSS 0
                THEN
                        BEGIN
                        IF .(LIST_ELEM_EXP1(.POINTER))<0, .VAL_SIZ*8, 1> NEQ .LIST_ELEM_EXP1(.POINTER)
                        THEN
                                SIGNAL(PAT$_NUMTRUNC);
                        END
                ELSE
                        IF .(LIST_ELEM_EXP1(.POINTER))<0, .VAL_SIZ*8, 0> NEQ .LIST_ELEM_EXP1(.POINTER)
                        THEN
                                SIGNAL(PAT$_NUMTRUNC);
                END;

        PAT$FILL_BUF(PAT$GL_TEMP_BUF, .OLD_VALUE_PTR, .VAL_SIZ);
        HOLE_SIZ = .HOLE_SIZ + .VAL_SIZ;                        ! Add current size to cumulative
                                                                ! Output old contents
        PAT$OUT_MEM_LOC(.NEXT_LOC, OLD_TAB_STG, PAT$GL_OLD_ASD, NO_CASE_TABLE);
        NEXT_LOC = .PAT$GL_NEXT_LOC;                            ! Point to next location
```

```
3
3               !++
3               ! Check if this is the last old argument.
3               !--
3               IF .LIST_ELEM_EXP2(.POINTER) EQL EXIT_TOKEN
3               THEN
3                       EXITLOOP;
2               END;
2
2       !++
2       ! Resolve any forward references in symbolic instruction operands.
2       !--
2       PAT$RESOLVE_INS(PAT$GL_TEMP_BUF);
2
2       !++
2       ! Now get the actual values in the locations and
2       ! check that they equal the specified values.
2       !--
2       BYTE_COUNT = 0;
3       WHILE (.BYTE_COUNT LSS .HOLE_SIZ)
2       DO
3               BEGIN
4               IF ((BUF_SIZE = .HOLE_SIZ - .BYTE_COUNT) GTR TTY_OUT_WIDTH)
3               THEN
3                       BUF_SIZE = TTY_OUT_WIDTH;
3               PAT$GET_VALUE(.LIST_ELEM_EXP1(.PAT$GL_HEAD_LST)+.BYTE_COUNT,
3                               .BUF_SIZE, OLD_CONTENTS);
3               IF CH$NEQ(.BUF_SIZE, .PAT$GL_TEMP_BUF[DSC$A_POINTER]+.BYTE_COUNT,
3                               .BUF_SIZE, OLD_CONTENTS)
3               THEN
3                       SIGNAL(PAT$_DIFVAL+MSG$K_WARN);
3               BYTE_COUNT = .BYTE_COUNT + .BUF_SIZE;
2               END;
2
2       !++
2       ! Release the storage holding the old instructions.
2       !--
2       PAT$FREERELEASE(.PAT$GL_TEMP_BUF[DSC$A_POINTER], (.PAT$GL_TEMP_BUF[DSC$W_LENGTH]+3)/4);
2       PAT$GL_TEMP_BUF[DSC$W_LENGTH] = 0;
2       PAT$GL_TEMP_BUF[DSC$A_POINTER] = 0;
2       !++
2       ! If this was a VERIFY command, we are all done.  Return for next command.
2       !--
2       IF .PAT$GL_CONTEXT[VERIFY_BIT]
2       THEN
2               RETURN;
2
2       !++
2       ! Check if old instruction should be moved to patch area, i.e., is this an
2       ! INSERT command.  Remember the number of bytes of old instructions moved
2       ! in case there are forward referenced symbols to relocate in the new
2       ! instructions.
2       !--
2       IF .PAT$GL_CONTEXT [INSERT_BIT]
2       THEN
3               BEGIN
3               PAT$FILL_BUF(PAT$GL_TEMP_BUF, INSTRUC_BUF[1], .INSTRUC_BUF[0]);
```

```
3               OLD_INS_SIZ = .PAT$GL_TEMP_BUF[DSC$W_LENGTH];           ! Remember # of bytes of old instructions mo
3               END
2       ELSE
3               BEGIN
3               OLD_INS_SIZ = 0;                                        ! No old instructions moved
3               NEXT_LOC = .UNMAPPED_LOC;                               ! Set next deposit location for REPLACE comm
2               END;
2
2       !++
2       ! Now fit the replacement value/instruction into the location.
2       !--
2       IF (NEW_INS_PTR = .LIST_ELEM_FLINK(.POINTER)) EQLA 0            ! If no replacement argument
2       THEN                                                            ! then report error
2               SIGNAL(PAT$_INVCMD);
2       PAT$GL_SYMTBPTR = .PAT$GL_NEWLABLS;                             ! Use the new contents label table
2
2       !++
2       ! Now build a buffer containing the new values to be deposited.  The deposits
2       ! are not done directly to memory in case part of the command is invalid.
2       !--
2       WHILE (POINTER = .LIST_ELEM_FLINK(.POINTER)) NEQA 0             ! Point to next argument
2       DO
3               BEGIN
3               IF .PAT$GB_MOD_PTR [MODE_INSTRUC]                       ! Test for instruction or data replacement
3               THEN
4                       BEGIN
4                       !++
4                       ! Now encode the replacement instruction.
4                       !--
4                       FILL_CHAR = NOP_INSTR;                          ! Set the fill character
4                       IF NOT PAT$INS_ENCODE(.LIST_ELEM_EXP1(.POINTER), INSTRUC_BUF,
4                                       .NEXT_LOC, PAT$GL_NEW_ASD, PAT$GL_TEMP_BUF)
4                       THEN
4                               SIGNAL(PAT$_NOENCODE, 1, .LIST_ELEM_EXP1(.POINTER));
4                       PAT$FILL_BUF(PAT$GL_TEMP_BUF, INSTRUC_BUF[1], .INSTRUC_BUF[0]); ! Insert instruction into te
4                       NEXT_LOC = .NEXT_LOC + .INSTRUC_BUF[0];
4                       END
3               ELSE
3                       !++
3                       ! The replacement is for a value.  Therefore it is on the parse
3                       ! stack.  Check for a truncation error.  Then set the fill
3                       ! character and write the value to the temporary buffer.
3                       !--
4                       BEGIN
4                       IF .LIST_ELEM_EXP1(.POINTER) LSS 0
4                       THEN
5                               BEGIN
5                               IF .(LIST_ELEM_EXP1(.POINTER))<0, .VAL_SIZ*8, 1> NEQ .LIST_ELEM_EXP1(.POINTER)
5                               THEN
5                                       SIGNAL(PAT$_NUMTRUNC);
5                               END
4                       ELSE
4                               IF .(LIST_ELEM_EXP1(.POINTER))<0, .VAL_SIZ*8, 0> NEQ .LIST_ELEM_EXP1(.POINTER)
4                               THEN
4                                       SIGNAL(PAT$_NUMTRUNC);
4                       FILL_CHAR = ZERO_BYTE;                          ! Set the fill character
4                       PAT$FILL_BUF(PAT$GL_TEMP_BUF, LIST_ELEM_EXP1(.POINTER), .VAL_SIZ);
```

```
4                       NEXT_LOC = .NEXT_LOC + .VAL_SIZ;
3                       END;
2               END;
2       !++
2       ! Resolve the forward references in symbolic instruction operands.
2       !--
2       PAT$RESOLVE_INS(PAT$GL_TEMP_BUF);
2
2       !++
2       ! Now check the replacement size against old instruction size.
2       !--
2       IF .PAT$GL_TEMP_BUF[DSC$W_LENGTH] LSS .HOLE_SIZ                 ! Make temporary buffer at least as large as
2       THEN
3               BEGIN
3               LOCAL
3                       TEMP_PTR;                                       ! Temporary pointer to temporary buffer
3
3               TEMP_PTR = PAT$FREEZ((.HOLE_SIZ + A_LONGWORD - 1)/A_LONGWORD);
3               CH$COPY(.PAT$GL_TEMP_BUF[DSC$W_LENGTH], .PAT$GL_TEMP_BUF[DSC$A_POINTER],
3                       .FILL_CHAR, .HOLE_SIZ, .TEMP_PTR);
3               PAT$FREERELEASE(.PAT$GL_TEMP_BUF[DSC$A_POINTER], (.PAT$GL_TEMP_BUF[DSC$W_LENGTH] +3)/4);
3               PAT$GL_TEMP_BUF[DSC$A_POINTER] = CH$PTR(.TEMP_PTR, 0);
3               PAT$GL_TEMP_BUF[DSC$W_LENGTH] = .HOLE_SIZ;
2               END;
2
2       !++
2       ! Now write the temporary buffer over the mapped input image.
2       !--
2       IF .PAT$GL_TEMP_BUF[DSC$W_LENGTH] EQL .HOLE_SIZ
2       THEN
3               BEGIN
3               !++
3               ! Replacement data fits.  Write it to memory and output new contents.
3               !--
3               PAT$WRITE_MEM(.UNMAPPED_LOC, .PAT$GL_TEMP_BUF[DSC$A_POINTER], .PAT$GL_TEMP_BUF[DSC$W_LENGTH]);
3               NEXT_LOC = .UNMAPPED_LOC + .HOLE_SIZ;
3               PAT$GL_NEXT_LOC = .UNMAPPED_LOC;
3               WHILE .PAT$GL_NEXT_LOC LSSA .NEXT_LOC                   ! Output new contents
3               DO
3                       PAT$OUT_MEM_LOC(.PAT$GL_NEXT_LOC, NEW_TAB_STG, PAT$GL_NEW_ASD, CASE_TABLE);
3               END
2       ELSE
3               BEGIN
3               !++
3               ! The replacement instruction is too large.  It
3               ! must be relocated to the patch area.
3               !--
3               IF .PAT$GB_MOD_PTR [MODE_INSTRUC]
3               THEN
3                       RELOCAT_INS(.UNMAPPED_LOC, .HOLE_SIZ, .OLD_INS_SIZ, .NEW_INS_PTR)
3               ELSE
3                       SIGNAL(PAT$_REPLACEERR);                        ! Internal error if patch area needed for da
2               END;
2       !++
2       ! Now add all the new labels to the user-defined symbol table.
2       !--
2       PAT$ADD_LABELS(PAT$GL_OLDLABLS);
```

```
; 1003          3657  2 PAT$ADD_LABELS(PAT$GL_NEWLABLS);
; 1004          3658  2 PAT$ADD_LABELS(PAT$GL_RLCLABLS);
; 1005          3659  2 RETURN;
; 1006          3660  1 END;                                                              ! End of PAT$REPLACE_CMD



                                                  0FFC 00000        .ENTRY  PAT$REPLACE_CMD, Save R2,R3,R4,R5,R6,R7,R8,-; 3315
                                                                            R9,R10,R11                                  ;
                               5E        FF2C  CE  9E 00002         MOVAB   -212(SP), SP                                ;
                               57   00000000G  EF  D0 00007         MOVL    PAT$GL_HEAD_LST, POINTER                    ; 3415
                                               04  13 0000E         BEQL    1$                                          ; 3416
                                               67  D5 00010         TSTL    (POINTER)                                   ;
                                               0D  12 00012         BNEQ    2$                                          ;
                                    006D80DA   8F  DD 00014 1$:     PUSHL   #7176410                                    ; 3418
                   00000000G   00              01  FB 0001A         CALLS   #1, LIB$SIGNAL                              ;
                   00000000G   EF              01  88 00021 2$:     BISB2   #1, PAT$GL_CONTEXT+1                        ; 3423
                               5B          04  A7  D0 00028         MOVL    4(POINTER), UNMAPPED_LOC                    ; 3424
                               59          04  A7  D0 0002C         MOVL    4(POINTER), NEXT_LOC                        ; 3425
                                               56  D4 00030         CLRL    HOLE_SIZ                                    ; 3426
                   00000000G   EF   00000000G  EF  D0 00032         MOVL    PAT$GL_OLDLABLS, PAT$GL_SYMTBPTR            ; 3427
                               57              67  D0 0003D 3$:     MOVL    (POINTER), POINTER                          ; 3435
                               52          04  A7  9E 00040         MOVAB   4(POINTER), R2                              ; 3451
                               50   00000000G  EF  D0 00044         MOVL    PAT$GB_MOD_PTR, R0                          ; 3441
                               38          03  A0  E9 0004B         BLBC    3(R0), 5$                                   ;
                                    00000000G  EF  9F 0004F         PUSHAB  PAT$GL_TEMP_BUF                             ; 3451
                                    00000000G  EF  9F 00055         PUSHAB  PAT$GL_OLD_ASD                              ;
                                               59  DD 0005B         PUSHL   NEXT_LOC                                    ; 3452
                                           B0  AD  9F 0005D         PUSHAB  INSTRUC_BUF                                 ; 3451
                                               62  DD 00060         PUSHL   (R2)                                        ;
                   00000000G   EF              05  FB 00062         CALLS   #5, PAT$INS_ENCODE                          ;
                               11              50  E8 00069         BLBS    R0, 4$                                      ;
                                               62  DD 0006C         PUSHL   (R2)                                        ; 3454
                                               01  DD 0006E         PUSHL   #1                                          ;
                                    006D810A   8F  DD 00070         PUSHL   #7176458                                    ;
                   00000000G   00              03  FB 00076         CALLS   #3, LIB$SIGNAL                              ;
                               53          B1  AD  9E 0007D 4$:     MOVAB   INSTRUC_BUF+1, OLD_VALUE_PTR                ; 3455
                               54          B0  AD  9A 00081         MOVZBL  INSTRUC_BUF, VAL_SIZ                        ; 3456
                                               2D  11 00085         BRB     8$                                          ; 3441
                               53              52  D0 00087 5$:     MOVL    R2, OLD_VALUE_PTR                           ; 3466
                               54          01  A0  9A 0008A         MOVZBL  1(R0), VAL_SIZ                              ; 3467
              50               54              03  78 0008E         ASHL    #3, VAL_SIZ, R0                             ; 3471
                                               62  D5 00092         TSTL    (R2)                                        ; 3468
                                               07  18 00094         BGEQ    6$                                          ;
    51        62               50              00  EE 00096         EXTV    #0, R0, (R2), R1                            ; 3471
                                               05  11 0009B         BRB     7$                                          ;
    51        62               50              00  EF 0009D 6$:     EXTZV   #0, R0, (R2), R1                            ; 3476
                               62              51  D1 000A2 7$:     CMPL    R1, (R2)                                    ;
                                               0D  13 000A5         BEQL    8$                                          ;
                                    006D8023   8F  DD 000A7         PUSHL   #7176227                                    ; 3478
                   00000000G   00              01  FB 000AD         CALLS   #1, LIB$SIGNAL                              ;
                                               18  BB 000B4 8$:     PUSHR   #^M<R3,R4>                                  ; 3481
                                    00000000G  EF  9F 000B6         PUSHAB  PAT$GL_TEMP_BUF                             ;
                   00000000V   EF              03  FB 000BC         CALLS   #3, PAT$FILL_BUF                            ;
                               56              54  C0 000C3         ADDL2   VAL_SIZ, HOLE_SIZ                           ; 3482
```

```
                              7E  D4 000C6          CLRL    -(SP)                                              ; 3484
               00000000G  EF  9F 000C8          PUSHAB  PAT$GL_OLD_ASD                                     ;
               00000000'  EF  9F 000CE          PUSHAB  OLD_TAB_STG                                        ;
                              59  DD 000D4          PUSHL   NEXT_LOC                                           ;
   00000000V  EF              04  FB 000D6          CALLS   #4, PAT$OUT_MEM_LOC                                ;
          59 00000000G        EF  D0 000DD          MOVL    PAT$GL_NEXT_LOC, NEXT_LOC                          ; 3485
          0A         08       A7  D1 000E4          CMPL    8(POINTER), #10                                    ; 3490
                              03  13 000E8          BEQL    9$                                                 ;
                            FF50  31 000EA          BRW     3$                                                 ;
               00000000G      EF  9F 000ED 9$:      PUSHAB  PAT$GL_TEMP_BUF                                    ; 3498
   00000000G  EF              01  FB 000F3          CALLS   #1, PAT$RESOLVE_INS                                ;
                              55  D4 000FA          CLRL    BYTE_COUNT                                         ; 3504
          56                  55  D1 000FC 10$:     CMPL    BYTE_COUNT, HOLE_SIZ                               ; 3505
                              44  18 000FF          BGEQ    13$                                                ;
58        56                  55  C3 00101          SUBL3   BYTE_COUNT, HOLE_SIZ, BUF_SIZE                     ; 3508
   00000084   8F              58  D1 00105          CMPL    BUF_SIZE, #132                                     ;
                              04  15 0010C          BLEQ    11$                                                ;
          58         84       8F  9A 0010E          MOVZBL  #132, BUF_SIZE                                     ; 3510
                   4100       8F  BB 00112 11$:     PUSHR   #^M<R8,SP>                                         ; 3512
          50 00000000G        EF  D0 00116          MOVL    PAT$GL_HEAD_LST, R0                                ; 3511
                     04 B045  9F     0011D          PUSHAB  @4(R0)[BYTE_COUNT]                                 ;
   00000000G  EF              03  FB 00121          CALLS   #3, PAT$GET_VALUE                                  ;
6E 00000000GFF45              58  29 00128          CMPC3   BUF_SIZE, @PAT$GL_TEMP_BUF+4[BYTE_COUNT], -        ; 3513
                                                            OLD_CONTENTS                                       ;
                              0D  13 00131          BEQL    12$                                                ;
               006D8290       8F  DD 00133          PUSHL   #7176848                                           ; 3516
   00000000G  00              01  FB 00139          CALLS   #1, LIB$SIGNAL                                     ;
          55                  58  C0 00140 12$:     ADDL2   BUF_SIZE, BYTE_COUNT                               ; 3517
                              B7  11 00143          BRB     10$                                                ; 3505
          50 00000000G        EF  3C 00145 13$:     MOVZWL  PAT$GL_TEMP_BUF, R0                                ; 3523
          50                  03  C0 0014C          ADDL2   #3, R0                                             ;
7E        50                  04  C7 0014F          DIVL3   #4, R0, -(SP)                                      ;
               00000000G      EF  DD 00153          PUSHL   PAT$GL_TEMP_BUF+4                                  ;
   00000000G  EF              02  FB 00159          CALLS   #2, PAT$FREERELEASE                                ;
               00000000G      EF  B4 00160          CLRW    PAT$GL_TEMP_BUF                                    ; 3524
               00000000G      EF  D4 00166          CLRL    PAT$GL_TEMP_BUF+4                                  ; 3525
01 00000000G  EF              05  E1 0016C          BBC     #5, PAT$GL_CONTEXT+2, 14$                          ; 3529
                                  04 00174          RET                                                        ;
               00000000G      EF  95 00175 14$:     TSTB    PAT$GL_CONTEXT+2                                   ; 3539
                              1D  18 0017B          BGEQ    15$                                                ;
          7E         B0       AD  9A 0017D          MOVZBL  INSTRUC_BUF, -(SP)                                 ; 3542
                     B1       AD  9F 00181          PUSHAB  INSTRUC_BUF+1                                      ;
               00000000G      EF  9F 00184          PUSHAB  PAT$GL_TEMP_BUF                                    ;
   00000000V  EF              03  FB 0018A          CALLS   #3, PAT$FILL_BUF                                   ;
          5A 00000000G        EF  3C 00191          MOVZWL  PAT$GL_TEMP_BUF, OLD_INS_SIZ                       ; 3543
                              05  11 00198          BRB     16$                                                ; 3539
                              5A  D4 0019A 15$:     CLRL    OLD_INS_SIZ                                        ; 3547
          59                  5B  D0 0019C          MOVL    UNMAPPED_LOC, NEXT_LOC                             ; 3548
          58                  67  D0 0019F 16$:     MOVL    (POINTER), NEW_INS_PTR                             ; 3554
                              0D  12 001A2          BNEQ    17$                                                ;
               006D80DA       8F  DD 001A4          PUSHL   #7176410                                           ; 3556
   00000000G  00              01  FB 001AA          CALLS   #1, LIB$SIGNAL                                     ;
   00000000G  EF 00000000G    EF  D0 001B1 17$:     MOVL    PAT$GL_NEWLABLS, PAT$GL_SYMTBPTR                   ; 3557
          57                  67  D0 001BC 18$:     MOVL    (POINTER), POINTER                                 ; 3563
                              03  12 001BF          BNEQ    19$                                                ;
                            0099  31 001C1          BRW     26$                                                ;
          52         04       A7  9E 001C4 19$:     MOVAB   4(POINTER), R2                                     ; 3573
```

```
                                 50 00000000G  EF   D0 001C8        MOVL    PAT$GB_MOD_PTR, R0                           ; 3566
                                 4E       03   A0   E9 001CF        BLBC    3(R0), 22$                                   ;
                                 53            01   90 001D3        MOVB    #1, FILL_CHAR                                ; 3572
                                    00000000G  EF   9F 001D6        PUSHAB  PAT$GL_TEMP_BUF                              ; 3573
                                    00000000G  EF   9F 001DC        PUSHAB  PAT$GL_NEW_ASD                               ;
                                               59   DD 001E2        PUSHL   NEXT_LOC                                     ; 3574
                                          B0   AD   9F 001E4        PUSHAB  INSTRUC_BUF                                  ; 3573
                                               62   DD 001E7        PUSHL   (R2)                                         ;
                     00000000G   EF            05   FB 001E9        CALLS   #5, PAT$INS_ENCODE                           ;
                                 11            50   E8 001F0        BLBS    R0, 20$                                      ;
                                               62   DD 001F3        PUSHL   (R2)                                         ; 3576
                                               01   DD 001F5        PUSHL   #1                                           ;
                                    006D810A   8F   DD 001F7        PUSHL   #7176458                                     ;
                     00000000G   00            03   FB 001FD        CALLS   #3, LIB$SIGNAL                               ;
                                 7E       B0   AD   9A 00204 20$:   MOVZBL  INSTRUC_BUF, -(SP)                           ; 3577
                                          B1   AD   9F 00208        PUSHAB  INSTRUC_BUF+1                                ;
                                    00000000G  EF   9F 0020B        PUSHAB  PAT$GL_TEMP_BUF                              ;
                     00000000V   EF            03   FB 00211        CALLS   #3, PAT$FILL_BUF                             ;
                                 50       B0   AD   9A 00218        MOVZBL  INSTRUC_BUF, R0                              ; 3578
                                 59            50   C0 0021C        ADDL2   R0, NEXT_LOC                                 ;
                                               9B   11 0021F 21$:   BRB     18$                                          ; 3566
                 50              54            03   78 00221 22$:   ASHL    #3, VAL_SIZ, R0                              ; 3590
                                               62   D5 00225        TSTL    (R2)                                         ; 3587
                                               07   18 00227        BGEQ    23$                                          ;
    51           62              50            00   EE 00229        EXTV    #0, R0, (R2), R1                             ; 3590
                                               05   11 0022E        BRB     24$                                          ;
    51           62              50            00   EF 00230 23$:   EXTZV   #0, R0, (R2), R1                             ; 3595
                                 62            51   D1 00235 24$:   CMPL    R1, (R2)                                     ;
                                               0D   13 00238        BEQL    25$                                          ;
                                    006D8023   8F   DD 0023A        PUSHL   #7176227                                     ; 3597
                     00000000G   00            01   FB 00240        CALLS   #1, LIB$SIGNAL                               ;
                                               53   94 00247 25$:   CLRB    FILL_CHAR                                    ; 3598
                                               14   BB 00249        PUSHR   #^M<R2,R4>                                   ; 3599
                                    00000000G  EF   9F 0024B        PUSHAB  PAT$GL_TEMP_BUF                              ;
                     00000000V   EF            03   FB 00251        CALLS   #3, PAT$FILL_BUF                             ;
                                 59            54   C0 00258        ADDL2   VAL_SIZ, NEXT_LOC                            ; 3600
                                               C2   11 0025B        BRB     21$                                          ; 3563
                                    00000000G  EF   9F 0025D 26$:   PUSHAB  PAT$GL_TEMP_BUF                              ; 3606
                     00000000G   EF            01   FB 00263        CALLS   #1, PAT$RESOLVE_INS                          ;
    56 00000000G  EF             10            00   ED 0026A        CMPZV   #0, #16, PAT$GL_TEMP_BUF, HOLE_SIZ           ; 3611
                                               49   18 00273        BGEQ    27$                                          ;
                                 50       03   A6   9E 00275        MOVAB   3(R6), R0                                    ; 3617
                 7E              50            04   C7 00279        DIVL3   #4, R0, -(SP)                                ;
                     00000000G   EF            01   FB 0027D        CALLS   #1, PAT$FREEZ                                ;
                                 57            50   D0 00284        MOVL    R0, TEMP_PTR                                 ;
    56           53  00000000G   FF 00000000G  EF   2C 00287        MOVC5   PAT$GL_TEMP_BUF, @PAT$GL_TEMP_BUF+4, -       ; 3619
                                               67      00294                FILL_CHAR, HOLE_SIZ, (TEMP_PTR)              ;
                                 50 00000000G  EF   3C 00295        MOVZWL  PAT$GL_TEMP_BUF, R0                          ; 3620
                                 50            03   C0 0029C        ADDL2   #3, R0                                       ;
                 7E              50            04   C7 0029F        DIVL3   #4, R0, -(SP)                                ;
                                    00000000G  EF   DD 002A3        PUSHL   PAT$GL_TEMP_BUF+4                            ;
                     00000000G   EF            02   FB 002A9        CALLS   #2, PAT$FREERELEASE                          ;
                     00000000G   EF            57   D0 002B0        MOVL    TEMP_PTR, PAT$GL_TEMP_BUF+4                  ; 3621
                     00000000G   EF            56   B0 002B7        MOVW    HOLE_SIZ, PAT$GL_TEMP_BUF                    ; 3622
    56 00000000G  EF             10            00   ED 002BE 27$:   CMPZV   #0, #16, PAT$GL_TEMP_BUF, HOLE_SIZ           ; 3628
                                               47   12 002C7        BNEQ    29$                                          ;
                                 7E 00000000G  EF   3C 002C9        MOVZWL  PAT$GL_TEMP_BUF, -(SP)                       ; 3634
```

```
                                000000000G  EF  DD 002D0           PUSHL   PAT$GL_TEMP_BUF+4                        ;
                                            5B  DD 002D6           PUSHL   UNMAPPED_LOC                             ;
                     00000000G  EF          03  FB 002D8           CALLS   #3, PAT$WRITE_MEM                        ;
                 59             5B          56  C1 002DF           ADDL3   HOLE_SIZ, UNMAPPED_LOC, NEXT_LOC         ; 3635
                     00000000G  EF          5B  D0 002E3           MOVL    UNMAPPED_LOC, PAT$GL_NEXT_LOC            ; 3636
                            59  00000000G   EF  D1 002EA 28$:      CMPL    PAT$GL_NEXT_LOC, NEXT_LOC                ; 3637
                                            46  1E 002F1           BGEQU   31$                                      ;
                                            01  DD 002F3           PUSHL   #1                                       ; 3639
                                000000000G  EF  9F 002F5           PUSHAB  PAT$GL_NEW_ASD                           ;
                                00000000'   EF  9F 002FB           PUSHAB  NEW_TAB_STG                              ;
                                000000000G  EF  DD 00301           PUSHL   PAT$GL_NEXT_LOC                          ;
                     00000000V  EF          04  FB 00307           CALLS   #4, PAT$OUT_MEM_LOC                      ;
                                            DA  11 0030E           BRB     28$                                      ;
                            50  00000000G   EF  D0 00310 29$:      MOVL    PAT$GB_MOD_PTR, R0                       ; 3647
                            11         03   A0  E9 00317           BLBC    3(R0), 30$                               ;
                                            58  DD 0031B           PUSHL   NEW_INS_PTR                              ; 3649
                                     0440   8F  BB 0031D           PUSHR   #^M<R6,R10>                              ;
                                            5B  DD 00321           PUSHL   UNMAPPED_LOC                             ;
                     00000000V  EF          04  FB 00323           CALLS   #4, RELOCAT_INS                          ;
                                            0D  11 0032A           BRB     31$                                      ;
                                006D815A    8F  DD 0032C 30$:      PUSHL   #7176538                                 ; 3651
                     00000000G  00          01  FB 00332           CALLS   #1, LIB$SIGNAL                           ;
                                00000000G   EF  9F 00339 31$:      PUSHAB  PAT$GL_OLDLABLS                          ; 3656
                     00000000G  EF          01  FB 0033F           CALLS   #1, PAT$ADD_LABELS                       ;
                                00000000G   EF  9F 00346           PUSHAB  PAT$GL_NEWLABLS                          ; 3657
                     00000000G  EF          01  FB 0034C           CALLS   #1, PAT$ADD_LABELS                       ;
                                00000000G   EF  9F 00353           PUSHAB  PAT$GL_RLCLABLS                          ; 3658
                     00000000G  EF          01  FB 00359           CALLS   #1, PAT$ADD_LABELS                       ;
                                                04 00360           RET                                              ; 3660

; Routine Size:  865 bytes,    Routine Base:  _PAT$CODE + 030A
```

```
ROUTINE RELOCAT_INS (OLD_LOC, HOLE_SIZE, OLD_INS_SIZ, ASC_INS_PTR) : NOVALUE =

!++
! FUNCTIONAL DESCRIPTION:
!
!       This routine relocates an instruction from an old address to the patch
!       area.  It then moves in any new instructions, specified as an argument
!       list for a patch command.  A branch or jump instruction is then put
!       into the old address.  If there is not enough room left by the
!       removal of the instruction, then more instructions are moved to the
!       patch area until the branch instruction iwll fit.  Lastly, a return
!       branch instruction is placed in the patch area to return execution
!       to the next sequential instruction past the old address.
!
!       Any new instructions to be inserted are in a command argument
!       list, created by the parser.  Each argument entry is made up of
!       three longwords. The first is a forward link to the next entry.
!       The second longword contains the address of a counted byte stream
!       that is to be translated into a binary instruction which is
!       to be inserted into the patch area.  The third longword is unused.
!
! CALLING SEQUENCE:
!
!       RELOCATE_CMD (OLD_LOCATION, NEW_INSTRUCTION_PTR)
!
! INPUTS:
!
!       OLD_LOC - Unmapped address of instruction to be moved
!       HOLE_SIZE - Number of free bytes at OLD_LOC
!       OLD_INS_SIZ - Number of bytes of old instruction preceding new instruction
!       ASC_INS_PTR - Pointer to first new instruction on command argument list
!
! IMPLICIT INPUTS:
!
!       PAT$GL_TEMP_BUF - String descriptor for counted binary instructions
!
!       The head of the linked list, the current mode, and
!       the current patch area descriptor.
!
! OUTPUTS:
!
!       none
!
! IMPLICIT OUTPUTS:
!
!       NONE
!
! ROUTINE VALUE:
!
!       novalue
!
! SIDE EFFECTS:
!
!       If the default patch area is to be used and it does not currently
!       exist when PAT$EXP_AREA is called PAT$BUILD_ISE is invoked which
!       given the address of the instructions to be moved will propagate
!       the image section attributes of the old image section to the newly
```

```
1 !        created default patch area image section descriptor.
1 !
1 !        The patch area now contains the moved instruction and the new ones
1 !        plus a branch instruction back to the inline code.  The old
1 !        instruction location contains a branch to the patch area.
1 !        If a failure in a write or contents verification occurs,
1 !        the routine returns immediately.
1 !
1 !--
1
2 BEGIN
2
2 LITERAL
2          MAX_BYTE_DISP = 127,                                    ! Maximum displacement for BRB
2          MIN_BYTE_DISP = -128,                                   ! Minimum displacement for BRB
2          MAX_WORD_DISP = 32767,                                  ! Maximum displacement for BRW
2          MIN_WORD_DISP = -32768,                                 ! Minimum displacement for BRW
2          BRB_OPCODE = %X'11',                                    ! Opcode for BRB
2          BRW_OPCODE = %X'31',                                    ! Opcode for BRW
2          JMP_OPCODE = %X'17',                                    ! Opcode for JMP
2          BRB_INS_SIZ = 2,                                        ! Size of BRB instruction
2          BRW_INS_SIZ = 3,                                        ! Size of BRW instruction
2          JMP_INS_SIZ = 6,                                        ! Size of JMP instruction
2          PC_DEFERRED = %X'EF',                                   ! PC deferred instruction mode
2          NOP_INSTR = 1,                                          ! Fill character for instruction
2          MAX_INST_LEN    = 80;                                   ! Maximum number of binary bytes in an instr
2
2 LOCAL
2          SUCC_OLD_INS,                                           ! Number of bytes of successive old instruct
2          DECODED_INS,                                            ! Pointer to ascii instruction
2          NXT_ASC_INS,                                            ! Pointer to next ascii instruction in argum
2          NEXT_PC,                                                ! PC of next instruction to decode
2          NEW_INS_PTR : REF VECTOR[,BYTE],                        ! Pointer to relocated instruction stream
2          BR_DISPLACEMENT : SIGNED LONG,                          ! Displacement for branch instruction
2          BR_INSTRUC : VECTOR[JMP_INS_SIZ+1,BYTE],                ! Encoded counted string branch instruction
2          NEW_LOC,                                                ! Address in patch area for relocated instru
2          NEXT_LOC,                                               ! Address of next instruction of inline code
2          CUR_LOC,                                                ! Address of current instruction to be moved
2          LOCAL_BUF : VECTOR[MAX_INST_LEN, BYTE],                 ! Local buffer for binary instruction stream
2          INSTRUC_BUF: VECTOR [MAX_INST_LEN, BYTE];               ! Local buffer for ascic instructions
2
2 !++
2 ! Enable instruction substitution.
2 !--
2 PAT$GL_CONTEXT[INST_SUBST] = TRUE;
2 PAT$GL_SYMTBPTR = .PAT$GL_RLCLABLS;
2
2 !++
2 ! Check that there is enough room in the patch area for the instructions
2 ! encoded in the temporary buffer, PAT$GL_TEMP_BUF.  This is the minimum size
2 ! that may be required.  Instruction substitution may enlarge this size.  This
2 ! will also insure that a patch area address is defined.
2 !--
3 IF (.PAT$GL_PATAREA[DSC$W_LENGTH] LSS .PAT$GL_TEMP_BUF[DSC$W_LENGTH])
2 THEN
3          BEGIN
4          IF (.PAT$GL_PATAREA[DSC$A_POINTER] EQLA .PAT$GL_IHPPTR[IHP$L_RW_PATADR])
```

```
                THEN
                        BEGIN
                        PAT$EXP_AREA((.PAT$GL_TEMP_BUF[DSC$W_LENGTH] + A_PAGE - 1)/A_PAGE, .OLD_LOC);
                        IF (.PAT$GL_PATAREA[DSC$W_LENGTH] LSS .PAT$GL_TEMP_BUF[DSC$W_LENGTH])
                        THEN
                                SIGNAL(PAT$_INSUFPAT, 3, .PAT$GL_TEMP_BUF[DSC$W_LENGTH],
                                  .PAT$GL_PATAREA[DSC$A_POINTER], .PAT$GL_PATAREA[DSC$W_LENGTH]);
                        END
                ELSE
                        SIGNAL(PAT$_INSUFPAT, 3, .PAT$GL_TEMP_BUF[DSC$W_LENGTH],
                         .PAT$GL_PATAREA[DSC$A_POINTER], .PAT$GL_PATAREA[DSC$W_LENGTH]);
                END;

        !++
        ! Set pointer to relocation address.
        !--
        NEW_LOC = CH$PTR(.PAT$GL_PATAREA[DSC$A_POINTER], 0);

        !++
        ! Now compute the branch displacement size.  Then build the binary code
        ! based on the displacement.
        !--
        BR_DISPLACEMENT = .NEW_LOC - .OLD_LOC - BRB_INS_SIZ;
        IF (.BR_DISPLACEMENT LEQ MAX_BYTE_DISP) AND (.BR_DISPLACEMENT GEQ MIN_BYTE_DISP)
        THEN
                BEGIN
                BR_INSTRUC[0] = BRB_INS_SIZ;
                BR_INSTRUC[1] = BRB_OPCODE;
                CH$MOVE(.BR_INSTRUC[0], CH$PTR(BR_DISPLACEMENT,0), CH$PTR(BR_INSTRUC[2],0));
                END
        ELSE
                IF (.BR_DISPLACEMENT LEQ MAX_WORD_DISP) AND (.BR_DISPLACEMENT GEQ MIN_WORD_DISP)
                THEN
                        BEGIN
                        BR_INSTRUC[0] = BRW_INS_SIZ;
                        BR_INSTRUC[1] = BRW_OPCODE;
                        BR_DISPLACEMENT = .BR_DISPLACEMENT - (BRW_INS_SIZ - BRB_INS_SIZ);
                        CH$MOVE(.BR_INSTRUC[0], CH$PTR(BR_DISPLACEMENT,0), CH$PTR(BR_INSTRUC[2],0));
                        END
                ELSE
                        BEGIN
                        BR_INSTRUC[0] = JMP_INS_SIZ;
                        BR_INSTRUC[1] = JMP_OPCODE;
                        BR_INSTRUC[2] = PC_DEFERRED;
                        BR_DISPLACEMENT = .BR_DISPLACEMENT - (JMP_INS_SIZ - BRB_INS_SIZ);
                        CH$MOVE(.BR_INSTRUC[0], CH$PTR(BR_DISPLACEMENT,0), CH$PTR(BR_INSTRUC[3],0));
                        END;

        !++
        ! Now see if the branch instruction will fit in the hole left at the old
        ! location.  If not, then move more instructions to the patch area until it
        ! will fit.
        !--
        NEXT_LOC = .OLD_LOC + .HOLE_SIZE;                               ! Compute address of next inline instruction
        NEXT_PC = .OLD_LOC + .HOLE_SIZE;                                ! Compute address of next inline instruction
        SUCC_OLD_INS = .PAT$GL_TEMP_BUF[DSC$W_LENGTH];                  ! Remember where extra old instructions move
        WHILE .BR_INSTRUC[0] GTR .HOLE_SIZE
```

```
2 DO
3         BEGIN
3         !++
3         ! First decode the instruction at the next old location.  Then
3         ! re-encode it to get the binary stream.  Then insert it into
3         ! the temporary buffer.
3         !--
3         PAT$OUT_MEM_LOC(.NEXT_LOC, OLD_TAB_STG, PAT$GL_OLD_ASD, NO_CASE_TABLE);
3         PAT$GL_BUF_SIZ = 0;
3         PAT$CP_OUT_STR = CH$PTR(LOCAL_BUF[1]);
3
3         !++
3         ! Because the instruction is actually in the code, the PC is the
3         ! same as the address of the byte stream.  PAT$INS_DECODE will update
3         ! the NEW_PC automatically.  To determine the length of the instruction
3         ! binary stream just decoded (for updating the HOLE_SIZE), NEXT_LOC must
3         ! be preserved.  Therefore, the return value from the routine is written
3         ! into NEW_PC not NEXT_LOC.  This will be the same value, unless the
3         ! routine failed.
3         !--
3         IF (NEXT_PC = PAT$INS_DECODE(.NEXT_LOC, 0, NEXT_PC, PAT$GL_OLD_ASD, NO_CASE_TABLE)) EQL 0
3         THEN
3                 SIGNAL(PAT$_NODECODE);
3         LOCAL_BUF[0] = .PAT$GL_BUF_SIZ;
3         NEW_INS_PTR = CH$PTR(INSTRUC_BUF, 0);
3         IF NOT PAT$INS_ENCODE(LOCAL_BUF, INSTRUC_BUF,
3                 .OLD_LOC + .PAT$GL_TEMP_BUF[DSC$W_LENGTH], PAT$GL_NEW_ASD, PAT$GL_TEMP_BUF)
3         THEN
4                 IF (.PAT$GB_SUBST_IN[0] NEQ 0)
3                 THEN
3                         NEW_INS_PTR = CH$PTR(PAT$GB_SUBST_IN, 0)
3                 ELSE
3                         SIGNAL(PAT$_NOENCODE, 1, LOCAL_BUF);
3         !++
3         ! There is a temporary restriction on relocation of CASE instructions
3         !--
3         IF (.NEW_INS_PTR[1] EQL OP_CASEB) OR
3            (.NEW_INS_PTR[1] EQL OP_CASEW) OR
4            (.NEW_INS_PTR[1] EQL OP_CASEL)
3         THEN
3                 SIGNAL(PAT$_NORELOC + MSG$K_SEVERE);
3         PAT$FILL_BUF(PAT$GL_TEMP_BUF, NEW_INS_PTR[1], .NEW_INS_PTR[0]);
3         HOLE_SIZE = .HOLE_SIZE + .NEXT_PC - .NEXT_LOC;
3         NEXT_LOC = .NEXT_PC;
2         END;
2
2 !++
2 ! Decode the instructions in the temporary buffer and re-encode them at the
2 ! patch area address.  This will alter the addresses within the instructions.
2 !--
2 CUR_LOC = .PAT$GL_TEMP_BUF[DSC$A_POINTER];
2 NEXT_PC = .OLD_LOC;
2 NXT_ASC_INS = .ASC_INS_PTR;                                    ! Get pointer to next new instruction argume
3 WHILE .CUR_LOC LSSA (.PAT$GL_TEMP_BUF[DSC$A_POINTER] + .PAT$GL_TEMP_BUF[DSC$W_LENGTH])
2 DO
3         BEGIN
3         PAT$GL_BUF_SIZ = 0;
```

```
        PAT$CP_OUT_STR = CH$PTR(LOCAL_BUF[1]);
        IF (.CUR_LOC GEQA .PAT$GL_TEMP_BUF[DSC$A_POINTER] + .OLD_INS_SIZ) AND
           (.CUR_LOC LSSA .PAT$GL_TEMP_BUF[DSC$A_POINTER] + .SUCC_OLD_INS)
        THEN
                BEGIN
                !++
                ! Take the new instructions out of the argument list in
                ! case there are any labels which will be relocated.
                ! This is only done for new instructions being deposited.
                ! The old instructions being relocated are decoded and re-encoded.
                !--
                DECODED_INS = .LIST_ELEM_EXP1(.NXT_ASC_INS);
                NXT_ASC_INS = .LIST_ELEM_FLINK(.NXT_ASC_INS);
                PAT$GL_SYMTBPTR = .PAT$GL_RLCLABLS;                     ! Use reolcated label table
                END
        ELSE
                BEGIN
                !++
                ! The instruction is an old instruction.  Therefore use the
                ! old label table and encode it from the decoded instruction.
                !--
                DECODED_INS = LOCAL_BUF;                                ! Point to ascii instruction
                PAT$GL_SYMTBPTR = .PAT$GL_OLDLABLS;                     ! Assume this is an old instruction
                END;
        IF (CUR_LOC = PAT$INS_DECODE(.CUR_LOC, 0, NEXT_PC, PAT$GL_NEW_ASD, NO_CASE_TABLE)) EQL 0
        THEN
                SIGNAL(PAT$_NODECODE);
        LOCAL_BUF[0] = .PAT$GL_BUF_SIZ;
        NEW_INS_PTR = CH$PTR(INSTRUC_BUF, 0);                           ! Set pointer to counted stream buffer
        IF NOT PAT$INS_ENCODE(.DECODED_INS, INSTRUC_BUF,
             .NEW_LOC + .PAT$GL_RLOC_BUF[DSC$W_LENGTH], PAT$GL_NEW_ASD, PAT$GL_RLOC_BUF)
        THEN
                IF (.PAT$GB_SUBST_IN[0] NEQ 0)
                THEN
                        NEW_INS_PTR = CH$PTR(PAT$GB_SUBST_IN, 0)
                ELSE
                        SIGNAL(PAT$_NOENCODE, 1, LOCAL_BUF);
        !++
        ! There is a temporary restriction on relocation of CASE instructions
        !--
        IF (.NEW_INS_PTR[1] EQL OP_CASEB) OR
           (.NEW_INS_PTR[1] EQL OP_CASEW) OR
           (.NEW_INS_PTR[1] EQL OP_CASEL)
        THEN
                SIGNAL(PAT$_NORELOC + MSG$K_SEVERE);
        PAT$FILL_BUF(PAT$GL_RLOC_BUF, NEW_INS_PTR[1], .NEW_INS_PTR[0]);
        END;
PAT$GL_SYMTBPTR = .PAT$GL_RLCLABLS;                                     ! Set relocated labels as default (old alway
PAT$RESOLVE_INS(PAT$GL_RLOC_BUF);

!++
! Now a return branch instruction must be placed in the relocation buffer.
! Compute the branch displacement size.  Then build the binary code based on
! the displacement.
!--
BR_DISPLACEMENT = .NEXT_LOC - (.PAT$GL_PATAREA[DSC$A_POINTER] +
                         .PAT$GL_RLOC_BUF[DSC$W_LENGTH]) - BRB_INS_SIZ;
```

```
3 IF (.BR_DISPLACEMENT LEQ MAX_BYTE_DISP) AND (.BR_DISPLACEMENT GEQ MIN_BYTE_DISP)
2 THEN
3         BEGIN
3         INSTRUC_BUF[0] = BRB_INS_SIZ;
3         INSTRUC_BUF[1] = BRB_OPCODE;
3         CH$MOVE(.INSTRUC_BUF[0], CH$PTR(BR_DISPLACEMENT,0), CH$PTR(INSTRUC_BUF[2],0));
3         END
2 ELSE
3         IF (.BR_DISPLACEMENT LEQ MAX_WORD_DISP) AND (.BR_DISPLACEMENT GEQ MIN_WORD_DISP)
2         THEN
3                 BEGIN
3                 INSTRUC_BUF[0] = BRW_INS_SIZ;
3                 INSTRUC_BUF[1] = BRW_OPCODE;
3                 BR_DISPLACEMENT = .BR_DISPLACEMENT - (BRW_INS_SIZ - BRB_INS_SIZ);
3                 CH$MOVE(.INSTRUC_BUF[0], CH$PTR(BR_DISPLACEMENT,0), CH$PTR(INSTRUC_BUF[2],0));
3                 END
2         ELSE
3                 BEGIN
3                 INSTRUC_BUF[0] = JMP_INS_SIZ;
3                 INSTRUC_BUF[1] = JMP_OPCODE;
3                 INSTRUC_BUF[2] = PC_DEFERRED;
3                 BR_DISPLACEMENT = .BR_DISPLACEMENT - (JMP_INS_SIZ - BRB_INS_SIZ);
3                 CH$MOVE(.INSTRUC_BUF[0], CH$PTR(BR_DISPLACEMENT,0), CH$PTR(INSTRUC_BUF[3],0));
2                 END;
2 PAT$FILL_BUF(PAT$GL_RLOC_BUF, INSTRUC_BUF[1], .INSTRUC_BUF[0]);
2
2 !++
2 ! Now insert all new instructions into the patch area.
2 !--
3 IF (.PAT$GL_RLOC_BUF[DSC$W_LENGTH] GTR .PAT$GL_PATAREA[DSC$W_LENGTH])
2 THEN
3         BEGIN
4         IF (.PAT$GL_PATAREA[DSC$A_POINTER] EQLA .PAT$GL_IHPPTR[IHP$L_RW_PATADR])
3         THEN
4                 BEGIN
4                 PAT$EXP_AREA((.PAT$GL_RLOC_BUF[DSC$W_LENGTH] + A_PAGE - 1)/A_PAGE, .OLD_LOC);
5                 IF (.PAT$GL_PATAREA[DSC$W_LENGTH] LSS .PAT$GL_RLOC_BUF[DSC$W_LENGTH])
4                 THEN
4                         SIGNAL(PAT$_INSUFPAT, 3, .PAT$GL_RLOC_BUF[DSC$W_LENGTH],
4                          .PAT$GL_PATAREA[DSC$A_POINTER], .PAT$GL_PATAREA[DSC$W_LENGTH]);
4                 END
3         ELSE
3                 SIGNAL(PAT$_INSUFPAT, 3, .PAT$GL_RLOC_BUF[DSC$W_LENGTH],
3                  .PAT$GL_PATAREA[DSC$A_POINTER], .PAT$GL_PATAREA[DSC$W_LENGTH]);
2         END;
2 PAT$WRITE_MEM(.PAT$GL_PATAREA[DSC$A_POINTER], .PAT$GL_RLOC_BUF[DSC$A_POINTER], .PAT$GL_RLOC_BUF[DSC$W_LENGTH
2 PAT$GL_PATAREA[DSC$W_LENGTH] = .PAT$GL_PATAREA[DSC$W_LENGTH] - .PAT$GL_RLOC_BUF[DSC$W_LENGTH];
2 PAT$GL_PATAREA[DSC$A_POINTER] = .PAT$GL_PATAREA[DSC$A_POINTER] + .PAT$GL_RLOC_BUF[DSC$W_LENGTH];
2
2 !++
2 ! Now there is room for the branch instruction at the old location hole.
2 ! Set up a buffer with the encoded branch instruction followed by NOP's to
2 ! insert there.  Then write it to the old location hole.
2 !--
3 IF (.HOLE_SIZE GTR .BR_INSTRUC[0])
2 THEN
3         BEGIN
```

```
: 1350          4003  3           NEW_INS_PTR = PAT$FREEZ((.HOLE_SIZE + A_LONGWORD - 1)/A_LONGWORD);
: 1351          4004  3           CH$COPY(.BR_INSTRUC[0], CH$PTR(BR_INSTRUC[1], 0), NOP_INSTR,
: 1352          4005  3                   .HOLE_SIZE, CH$PTR(.NEW_INS_PTR, 0));
: 1353          4006  3           PAT$WRITE_MEM(.OLD_LOC, CH$PTR(.NEW_INS_PTR, 0), .HOLE_SIZE);
: 1354          4007  3           PAT$FREERELEASE(CH$PTR(.NEW_INS_PTR, 0), (.HOLE_SIZE + 3)/4);
: 1355          4008  3           END
: 1356          4009  2   ELSE
: 1357          4010  2           PAT$WRITE_MEM(.OLD_LOC, CH$PTR(BR_INSTRUC[1], 0), .HOLE_SIZE);
: 1358          4011  2
: 1359          4012  2   !++
: 1360          4013  2   ! Now write out all the new instructions deposited.
: 1361          4014  2   !--
: 1362          4015  2   NEXT_LOC = .OLD_LOC;
: 1363          4016  3   WHILE (.NEXT_LOC LSS .OLD_LOC+.HOLE_SIZE)
: 1364          4017  2   DO
: 1365          4018  3           BEGIN
: 1366          4019  3           PAT$OUT_MEM_LOC(.NEXT_LOC, NEW_TAB_STG, PAT$GL_NEW_ASD, NO_CASE_TABLE);
: 1367          4020  3           NEXT_LOC = .PAT$GL_NEXT_LOC;
: 1368          4021  2           END;
: 1369          4022  2   NEXT_LOC = .NEW_LOC;
: 1370          4023  3   WHILE (.NEXT_LOC LSS .PAT$GL_PATAREA[DSC$A_POINTER])
: 1371          4024  2   DO
: 1372          4025  3           BEGIN
: 1373          4026  3           PAT$OUT_MEM_LOC(.NEXT_LOC, NEW_TAB_STG, PAT$GL_NEW_ASD, NO_CASE_TABLE);
: 1374          4027  3           NEXT_LOC = .PAT$GL_NEXT_LOC;
: 1375          4028  2           END;
: 1376          4029  2
: 1377          4030  2   RETURN;
: 1378          4031  1   END;                                                              ! End of RELOCAT_INS



                                      0FFC 00000 RELOCAT_INS:
                                                          .WORD    Save R2,R3,R4,R5,R6,R7,R8,R9,R10,R11          ; 3661
                    5B 0000000G   EF   9E 00002           MOVAB    PAT$GL_RLOC_BUF, R11                          ;
                    5E     FF50   CE   9E 00009           MOVAB    -176(SP), SP                                  ;
          0000000G  EF            10   88 0000E           BISB2    #16, PAT$GL_CONTEXT+2                         ; 3762
          0000000G  EF 0000000G   EF   D0 00015           MOVL     PAT$GL_RLCLABLS, PAT$GL_SYMTBPTR              ; 3763
                    50 0000000G   EF   D0 00020           MOVL     PAT$GL_PATAREA, R0                            ; 3771
                    52 0000000G   EF   3C 00027           MOVZWL   PAT$GL_TEMP_BUF, R2                           ;
                    52            60   B1 0002E           CMPW     (R0), R2                                      ;
                                  5B   1E 00031           BGEQU    3$                                            ;
                    51 0000000G   EF   D0 00033           MOVL     PAT$GL_IHPPTR, R1                             ; 3774
              14    A1       04   A0   D1 0003A           CMPL     4(R0), 20(R1)                                 ;
                                  36   12 0003F           BNEQ     1$                                            ;
                             04   AC   DD 00041           PUSHL    OLD_LOC                                       ; 3777
                    52     01FF   C2   9E 00044           MOVAB    511(R2), R2                                   ;
       7E           52 00000200   8F   C7 00049           DIVL3    #512, R2, -(SP)                               ;
          0000000G  EF            02   FB 00051           CALLS    #2, PAT$EXP_AREA                              ;
                    50 0000000G   EF   D0 00058           MOVL     PAT$GL_PATAREA, R0                            ; 3778
          0000000G  EF            60   B1 0005F           CMPW     (R0), PAT$GL_TEMP_BUF                         ;
                                  26   1E 00066           BGEQU    3$                                            ;
                    7E            60   3C 00068           MOVZWL   (R0), -(SP)                                   ; 3781
                             04   A0   DD 0006B           PUSHL    4(R0)                                         ;
                    7E 0000000G   EF   3C 0006E           MOVZWL   PAT$GL_TEMP_BUF, -(SP)                        ; 3780
```

```
                                                          L 6
PATEXA                                           16-Sep-1984 00:30:29    VAX-11 Bliss-32 V4.0-742              Page  39
V04-000                                          14-Sep-1984 12:52:32    DISK$VMSMASTER:[PATCH.SRC]PATEXA.B32;1  (7)

                                              08   11 00075         BRB     2$                                    ;
                        7E                    60   3C 00077  1$:    MOVZWL  (R0), -(SP)                           ; 3785
                                       04     A0   DD 0007A         PUSHL   4(R0)                                 ;
                                              52   DD 0007D         PUSHL   R2                                    ; 3784
                                              03   DD 0007F  2$:    PUSHL   #3                                    ;
                              006D80C2        8F   DD 00081         PUSHL   #7176386                              ;
                  00000000G   00              05   FB 00087         CALLS   #5, LIB$SIGNAL                        ;
                              50 00000000G    EF   D0 0008E  3$:    MOVL    PAT$GL_PATAREA, R0                    ; 3791
                              56       04     A0   D0 00095         MOVL    4(R0), NEW_LOC                        ;
                              59       04     AC   D0 00099         MOVL    OLD_LOC, R9                           ; 3797
            50                56              59   C3 0009D         SUBL3   R9, NEW_LOC, R0                       ;
                       04     AE       FE     A0   9E 000A1         MOVAB   -2(R0), BR_DISPLACEMENT               ;
                  0000007F    8F       04     AE   D1 000A6         CMPL    BR_DISPLACEMENT, #127                 ; 3798
                                              12   14 000AE         BGTR    4$                                    ;
                  FFFFFF80    8F       04     AE   D1 000B0         CMPL    BR_DISPLACEMENT, #-128                ;
                                              08   19 000B8         BLSS    4$                                    ;
                       F8     AD     1102     8F   B0 000BA         MOVW    #4354, BR_INSTRUC                     ; 3801
                                              1D   11 000C0         BRB     5$                                    ; 3803
                  00007FFF    8F       04     AE   D1 000C2  4$:    CMPL    BR_DISPLACEMENT, #32767               ; 3806
                                              1F   14 000CA         BGTR    6$                                    ;
                  FFFF8000    8F       04     AE   D1 000CC         CMPL    BR_DISPLACEMENT, #-32768              ;
                                              15   19 000D4         BLSS    6$                                    ;
                       F8     AD     3103     8F   B0 000D6         MOVW    #12547, BR_INSTRUC                    ; 3809
                                       04     AE   D7 000DC         DECL    BR_DISPLACEMENT                       ; 3811
                              50       F8     AD   9A 000DF  5$:    MOVZBL  BR_INSTRUC, R0                        ; 3812
        FA    AD       04     AE              50   28 000E3         MOVC3   R0, BR_DISPLACEMENT, BR_INSTRUC+2     ;
                                              18   11 000E9         BRB     7$                                    ; 3806
                       F8     AD     1706     8F   B0 000EB  6$:    MOVW    #5894, BR_INSTRUC                     ; 3816
                       FA     AD              11   8E 000F1         MNEGB   #17, BR_INSTRUC+2                     ; 3818
                       04     AE              04   C2 000F5         SUBL2   #4, BR_DISPLACEMENT                   ; 3819
                              50       F8     AD   9A 000F9         MOVZBL  BR_INSTRUC, R0                        ; 3820
        FB    AD       04     AE              50   28 000FD         MOVC3   R0, BR_DISPLACEMENT, BR_INSTRUC+3     ;
            50                59       08     AC   C1 00103  7$:    ADDL3   HOLE_SIZE, R9, R0                     ; 3828
                              5A              50   D0 00108         MOVL    R0, NEXT_LOC                          ;
                              6E              50   D0 0010B         MOVL    R0, NEXT_PC                           ; 3829
                              55 00000000G    EF   3C 0010E         MOVZWL  PAT$GL_TEMP_BUF, SUCC_OLD_INS         ; 3830
08   AC         F8    AD                      08   00 ED 00115  8$: CMPZV   #0, #8, BR_INSTRUC, HOLE_SIZE         ; 3831
                                              03   14 0011C         BGTR    9$                                    ;
                                            00E7   31 0011E         BRW     15$                                   ;
                                              7E   D4 00121  9$:    CLRL    -(SP)                                 ; 3839
                              00000000G       EF   9F 00123         PUSHAB  PAT$GL_OLD_ASD                        ;
                              00000000'       EF   9F 00129         PUSHAB  OLD_TAB_STG                           ;
                                              5A   DD 0012F         PUSHL   NEXT_LOC                              ;
                  00000000V   EF              04   FB 00131         CALLS   #4, PAT$OUT_MEM_LOC                   ;
                              00000000G       EF   D4 00138         CLRL    PAT$GL_BUF_SIZ                        ; 3840
                  00000000G   EF       59     AE   9E 0013E         MOVAB   LOCAL_BUF+1, PAT$CP_OUT_STR           ; 3841
                                              7E   D4 00146         CLRL    -(SP)                                 ; 3852
                              00000000G       EF   9F 00148         PUSHAB  PAT$GL_OLD_ASD                        ;
                                       08     AE   9F 0014E         PUSHAB  NEXT_PC                               ;
                                              7E   D4 00151         CLRL    -(SP)                                 ;
                                              5A   DD 00153         PUSHL   NEXT_LOC                              ;
                  00000000G   EF              05   FB 00155         CALLS   #5, PAT$INS_DECODE                    ;
                              6E              50   D0 0015C         MOVL    R0, NEXT_PC                           ;
                                              0D   12 0015F         BNEQ    10$                                   ;
                              006D8102        8F   DD 00161         PUSHL   #7176450                              ; 3854
                  00000000G   00              01   FB 00167         CALLS   #1, LIB$SIGNAL                        ;
                       58     AE 00000000G    EF   90 0016E  10$:   MOVB    PAT$GL_BUF_SIZ, LOCAL_BUF             ; 3855
```

```
                         57        08  AE  9E 00176        MOVAB   INSTRUC_BUF, NEW_INS_PTR                     ; 3856
                            00000000G  EF  9F 0017A        PUSHAB  PAT$GL_TEMP_BUF                              ; 3857
                            00000000G  EF  9F 00180        PUSHAB  PAT$GL_NEW_ASD                               ;
                         50 00000000G  EF  3C 00186        MOVZWL  PAT$GL_TEMP_BUF, R0                          ; 3858
                                     6049  9F 0018D        PUSHAB  (R0)[R9]                                     ;
                                   14  AE  9F 00190        PUSHAB  INSTRUC_BUF                                  ; 3857
                                   68  AE  9F 00193        PUSHAB  LOCAL_BUF                                    ;
             00000000G  EF             05  FB 00196        CALLS   #5, PAT$INS_ENCODE                           ;
                        23             50  E8 0019D        BLBS    R0, 12$                                      ;
                            00000000G  EF  95 001A0        TSTB    PAT$GB_SUBST_IN                              ; 3860
                                   09  13 001A6            BEQL    11$                                          ;
                         57 00000000G  EF  9E 001A8        MOVAB   PAT$GB_SUBST_IN, NEW_INS_PTR                 ; 3862
                                   12  11 001AF            BRB     12$                                          ;
                                   58  AE  9F 001B1  11$:  PUSHAB  LOCAL_BUF                                    ; 3864
                                   01  DD 001B4            PUSHL   #1                                           ;
                             006D810A  8F  DD 001B6        PUSHL   #7176458                                     ;
             00000000G  00             03  FB 001BC        CALLS   #3, LIB$SIGNAL                               ;
                    8F  8F         01  A7  91 001C3  12$:  CMPB    1(NEW_INS_PTR), #143                         ; 3868
                                   0E  13 001C8            BEQL    13$                                          ;
                    AF  8F         01  A7  91 001CA        CMPB    1(NEW_INS_PTR), #175                         ; 3869
                                   07  13 001CF            BEQL    13$                                          ;
                    CF  8F         01  A7  91 001D1        CMPB    1(NEW_INS_PTR), #207                         ; 3870
                                   0D  12 001D6            BNEQ    14$                                          ;
                             006D82CA  8F  DD 001D8  13$:  PUSHL   #7176906                                     ; 3872
             00000000G  00             01  FB 001DE        CALLS   #1, LIB$SIGNAL                               ;
                        7E             67  9A 001E5  14$:  MOVZBL  (NEW_INS_PTR), -(SP)                         ; 3873
                                   01  A7  9F 001E8        PUSHAB  1(NEW_INS_PTR)                               ;
                            00000000G  EF  9F 001EB        PUSHAB  PAT$GL_TEMP_BUF                              ;
             00000000V  EF             03  FB 001F1        CALLS   #3, PAT$FILL_BUF                             ;
         50        08  AC              6E  C1 001F8        ADDL3   NEXT_PC, HOLE_SIZE, R0                       ; 3874
   08    AC            50              5A  C3 001FD        SUBL3   NEXT_LOC, R0, HOLE_SIZE                      ;
                        5A             6E  D0 00202        MOVL    NEXT_PC, NEXT_LOC                            ; 3875
                                     FF0D  31 00205        BRW     8$                                           ; 3831
                         53 00000000G  EF  D0 00208  15$:  MOVL    PAT$GL_TEMP_BUF+4, CUR_LOC                   ; 3882
                         6E            59  D0 0020F        MOVL    R9, NEXT_PC                                  ; 3883
                         52        10  AC  D0 00212        MOVL    ASC_INS_PTR, NXT_ASC_INS                     ; 3884
                         50 00000000G  EF  D0 00216  16$:  MOVL    PAT$GL_TEMP_BUF+4, R0                        ; 3885
                         51 00000000G  EF  3C 0021D        MOVZWL  PAT$GL_TEMP_BUF, R1                          ;
                         51            50  C0 00224        ADDL2   R0, R1                                       ;
                         51            53  D1 00227        CMPL    CUR_LOC, R1                                  ;
                                   03  1F 0022A            BLSSU   17$                                          ;
                                     00EB  31 0022C        BRW     25$                                          ;
                            00000000G  EF  D4 0022F  17$:  CLRL    PAT$GL_BUF_SIZ                               ; 3888
             00000000G  EF         59  AE  9E 00235        MOVAB   LOCAL_BUF+1, PAT$CP_OUT_STR                  ; 3889
         51             50         0C  AC  C1 0023D        ADDL3   OLD_INS_SIZ, R0, R1                          ; 3890
                        51             53  D1 00242        CMPL    CUR_LOC, R1                                  ;
                                   1C  1F 00245            BLSSU   18$                                          ;
                        50             55  C0 00247        ADDL2   SUCC_OLD_INS, R0                             ; 3891
                        50             53  D1 0024A        CMPL    CUR_LOC, R0                                  ;
                                   14  1E 0024D            BGEQU   18$                                          ;
                        54         04  A2  D0 0024F        MOVL    4(NXT_ASC_INS), DECODED_INS                  ; 3900
                        52             62  D0 00253        MOVL    (NXT_ASC_INS), NXT_ASC_INS                   ; 3901
             00000000G  EF  00000000G  EF  D0 00256        MOVL    PAT$GL_RECLABLS, PAT$GL_SYMTBPTR             ; 3902
                                   0F  11 00261            BRB     19$                                          ; 3890
                        54         58  AE  9E 00263  18$:  MOVAB   LOCAL_BUF, DECODED_INS                       ; 3910
             00000000G  EF  00000000G  EF  D0 00267        MOVL    PAT$GL_OLDLABLS, PAT$GL_SYMTBPTR             ; 3911
                                   7E  D4 00272      19$:  CLRL    -(SP)                                        ; 3913
```

```
                                                   N 6
PATEXA                                             16-Sep-1984 00:30:29    VAX-11 Bliss-32 V4.0-742                    Page  41
V04-000                                            14-Sep-1984 12:52:32    DISK$VMSMASTER:[PATCH.SRC]PATEXA.B32;1       (7)

                  00000000G  EF  9F 00274          PUSHAB  PAT$GL_NEW_ASD                          ;
                         08  AE  9F 0027A          PUSHAB  NEXT_PC                                 ;
                             7E  D4 0027D          CLRL    -(SP)                                   ;
                             53  DD 0027F          PUSHL   CUR_LOC                                 ;
       00000000G  EF         05  FB 00281          CALLS   #5, PAT$INS_DECODE                      ;
                  53         50  D0 00288          MOVL    R0, CUR_LOC                             ;
                             0D  12 0028B          BNEQ    20$                                     ;
                  006D8102   8F  DD 0028D          PUSHL   #7176450                                ; 3915
       00000000G  00         01  FB 00293          CALLS   #1, LIB$SIGNAL                          ;
              58  AE 00000000G  EF  90 0029A 20$:  MOVB    PAT$GL_BUF_SIZ, LOCAL_BUF               ; 3916
                  57         08  AE  9E 002A2      MOVAB   INSTRUC_BUF, NEW_INS_PTR                ; 3917
                             5B  DD 002A6          PUSHL   R11                                     ; 3918
                  00000000G  EF  9F 002A8          PUSHAB  PAT$GL_NEW_ASD                          ;
                  50         6B  3C 002AE          MOVZWL  PAT$GL_RLOC_BUF, R0                     ; 3919
                           6046  9F 002B1          PUSHAB  (R0)[NEW_LOC]                           ;
                         14  AE  9F 002B4          PUSHAB  INSTRUC_BUF                             ; 3918
                             54  DD 002B7          PUSHL   DECODED_INS                             ;
       00000000G  EF         05  FB 002B9          CALLS   #5, PAT$INS_ENCODE                      ;
                  23         50  E8 002C0          BLBS    R0, 22$                                 ;
                  00000000G  EF  95 002C3          TSTB    PAT$GB_SUBST_IN                         ; 3921
                             09  13 002C9          BEQL    21$                                     ;
                  57 00000000G  EF  9E 002CB       MOVAB   PAT$GB_SUBST_IN, NEW_INS_PTR            ; 3923
                             12  11 002D2          BRB     22$                                     ;
                         58  AE  9F 002D4 21$:     PUSHAB  LOCAL_BUF                               ; 3925
                             01  DD 002D7          PUSHL   #1                                      ;
                  006D810A   8F  DD 002D9          PUSHL   #7176458                                ;
       00000000G  00         03  FB 002DF          CALLS   #3, LIB$SIGNAL                          ;
              8F  8F     01  A7  91 002E6 22$:     CMPB    1(NEW_INS_PTR), #143                    ; 3929
                             0E  13 002EB          BEQL    23$                                     ;
              AF  8F     01  A7  91 002ED          CMPB    1(NEW_INS_PTR), #175                    ; 3930
                             07  13 002F2          BEQL    23$                                     ;
              CF  8F     01  A7  91 002F4          CMPB    1(NEW_INS_PTR), #207                    ; 3931
                             0D  12 002F9          BNEQ    24$                                     ;
                  006D82CA   8F  DD 002FB 23$:     PUSHL   #7176906                                ; 3933
       00000000G  00         01  FB 00301          CALLS   #1, LIB$SIGNAL                          ;
                  7E         67  9A 00308 24$:     MOVZBL  (NEW_INS_PTR), -(SP)                    ; 3934
                         01  A7  9F 0030B          PUSHAB  1(NEQ_INS_PTR)                          ;
                             5B  DD 0030E          PUSHL   R11                                     ;
       00000000V  EF         03  FB 00310          CALLS   #3, PAT$FILL_BUF                        ;
                           FEFC  31 00317          BRW     16$                                     ; 3885
       00000000G  EF 00000000G  EF  D0 0031A 25$:  MOVL    PAT$GL_RLCLABLS, PAT$GL_SYMTBPTR        ; 3936
                             5B  DD 00325          PUSHL   R11                                     ; 3937
       00000000G  EF         01  FB 00327          CALLS   #1, PAT$RESOLVE_INS                     ;
                  50 00000000G  EF  D0 0032E       MOVL    PAT$GL_PATAREA, R0                      ; 3944
                  51         6B  3C 00335          MOVZWL  PAT$GL_RLOC_BUF, R1                     ; 3945
50                51     04  A0  C1 00338          ADDL3   4(R0), R1, R0                           ;
                  50         5A  C2 0033D          SUBL2   NEXT_LOC, R0                            ; 3944
              04  AE         50  CE 00340          MNEGL   R0, BR_DISPLACEMENT                     ; 3945
              04  AE         02  C2 00344          SUBL2   #2, BR_DISPLACEMENT                     ;
       0000007F   8F     04  AE  D1 00348          CMPL    BR_DISPLACEMENT, #127                   ; 3946
                             12  14 00350          BGTR    26$                                     ;
       FFFFFF80   8F     04  AE  D1 00352          CMPL    BR_DISPLACEMENT, #-128                  ;
                             08  19 0035A          BLSS    26$                                     ;
              08  AE   1102  8F  B0 0035C          MOVW    #4354, INSTRUC_BUF                      ; 3949
                             1D  11 00362          BRB     27$                                     ; 3951
       00007FFF   8F     04  AE  D1 00364 26$:     CMPL    BR_DISPLACEMENT, #32767                 ; 3954
                             1F  14 0036C          BGTR    28$                                     ;
```

```
                                                                 B  7
PATEXA                                                     16-Sep-1984 00:30:29     VAX-11 Bliss-32 V4.0-742                 Page  42
V04-000                                                    14-Sep-1984 12:52:32     DISK$VMSMASTER:[PATCH.SRC]PATEXA.B32;1    (7)

                       FFFF8000  8F        04  AE  D1 0036E        CMPL     BR_DISPLACEMENT, #-32768                  :
                                               15  19 00376        BLSS     28$                                       :
                             08  AE      3103  8F  B0 00378        MOVW     #12547, INSTRUC_BUF                       : 3957
                                           04  AE  D7 0037E        DECL     BR_DISPLACEMENT                           : 3959
                             50        08  AE  9A 00381 27$:       MOVZBL   INSTRUC_BUF, R0                           : 3960
              0A  AE         04  AE            50  28 00385        MOVC3    R0, BR_DISPLACEMENT, INSTRUC_BUF+2        :
                                               18  11 0038B        BRB      29$                                       : 3954
                             08  AE      1706  8F  B0 0038D 28$:   MOVW     #5894, INSTRUC_BUF                        : 3964
                             0A  AE            11  8E 00393        MNEGB    #17, INSTRUC_BUF+2                        : 3966
                             04  AE            04  C2 00397        SUBL2    #4, BR_DISPLACEMENT                       : 3967
                             50        08  AE  9A 0039B        MOVZBL   INSTRUC_BUF, R0                           : 3968
              0B  AE         04  AE            50  28 0039F        MOVC3    R0, BR_DISPLACEMENT, INSTRUC_BUF+3        :
                                 7E        08  AE  9A 003A5 29$:   MOVZBL   INSTRUC_BUF, -(SP)                        : 3970
                                           0D  AE  9F 003A9        PUSHAB   INSTRUC_BUF+1                             :
                                               5B  DD 003AC        PUSHL    R11                                       :
                       00000000V  EF               03  FB 003AE        CALLS    #3, PAT$FILL_BUF                          :
                                 52               6B  3C 003B5        MOVZWL   PAT$GL_RLOC_BUF, R2                       : 3975
                                 50 00000000G  EF  D0 003B8        MOVL     PAT$GL_PATAREA, R0                        :
                                 52               60  B1 003BF        CMPW     (R0), R2                                  :
                                                  52  1E 003C2        BGEQU    32$                                       :
                                 51 00000000G  EF  D0 003C4        MOVL     PAT$GL_IHPPTR, R1                         : 3978
                             14  A1        04  A0  D1 003CB        CMPL     4(R0), 20(R1)                             :
                                                  2D  12 003D0        BNEQ     30$                                       :
                                                  59  DD 003D2        PUSHL    R9                                        : 3981
                                 52      01FF  C2  9E 003D4        MOVAB    511(R2), R2                               :
                  7E             52 00000200  8F  C7 003D9        DIVL3    #512, R2, -(SP)                           :
                       00000000G  EF               02  FB 003E1        CALLS    #2, PAT$EXP_AREA                          :
                                 50 00000000G  EF  D0 003E8        MOVL     PAT$GL_PATAREA, R0                        : 3982
                                 6B               60  B1 003EF        CMPW     (R0), PAT$GL_RLOC_BUF                     :
                                                  22  1E 003F2        BGEQU    32$                                       :
                                 7E               60  3C 003F4        MOVZWL   (R0), -(SP)                               : 3985
                                           04  A0  DD 003F7        PUSHL    4(R0)                                     :
                                 7E               6B  3C 003FA        MOVZWL   PAT$GL_RLOC_BUF, -(SP)                    : 3984
                                                  08  11 003FD        BRB      31$                                       :
                                 7E               60  3C 003FF 30$:   MOVZWL   (R0), -(SP)                               : 3989
                                           04  A0  DD 00402        PUSHL    4(R0)                                     :
                                                  52  DD 00405        PUSHL    R2                                        : 3988
                                                  03  DD 00407 31$:   PUSHL    #3                                        :
                                    006D80C2  8F  DD 00409        PUSHL    #7176386                                  :
                       00000000G  00               05  FB 0040F        CALLS    #5, LIB$SIGNAL                            :
                                 7E               6B  3C 00416 32$:   MOVZWL   PAT$GL_RLOC_BUF, -(SP)                    : 3991
                                           04  AB  DD 00419        PUSHL    PAT$GL_RLOC_BUF+4                         :
                                 50 00000000G  EF  D0 0041C        MOVL     PAT$GL_PATAREA, R0                        :
                                           04  A0  DD 00423        PUSHL    4(R0)                                     :
                       00000000G  EF               03  FB 00426        CALLS    #3, PAT$WRITE_MEM                         :
                                 50 00000000G  EF  D0 0042D        MOVL     PAT$GL_PATAREA, R0                        : 3992
                                 60               6B  A2 00434        SUBW2    PAT$GL_RLOC_BUF, (R0)                     :
                                 51               6B  3C 00437        MOVZWL   PAT$GL_RLOC_BUF, R1                       : 3993
                             04  A0               51  C0 0043A        ADDL2    R1, 4(R0)                                 :
                                 58        08  AC  D0 0043E        MOVL     HOLE_SIZE, R8                             : 4000
      58          F8  AD         08               00  ED 00442        CMPZV    #0, #8, BR_INSTRUC, R8                    :
                                 3C               18 00448        BGEQ     33$                                       :
                                 50        03  A8  9E 0044A        MOVAB    3(R8), R0                                 : 4003
                  7E             50               04  C7 0044E        DIVL3    #4, R0, -(SP)                             :
                       00000000G  EF               01  FB 00452        CALLS    #1, PAT$FREEZ                             :
                                 57               50  D0 00459        MOVL     R0, NEW_INS_PTR                           :
                                 50        F8  AD  9A 0045C        MOVZBL   BR_INSTRUC, R0                            : 4004
```

```
          58          01       F9  AD          50  2C 00460             MOVC5   R0, BR_INSTRUC+1, #1, R8, (NEW_INS_PTR)       ; 4005
                                               67     00466                                                                   ;
                                   7E          57  7D 00467             MOVQ    NEW_INS_PTR, -(SP)                             ; 4006
                                               59  DD 0046A             PUSHL   R9                                             ;
                        00000000G  EF          03  FB 0046C             CALLS   #3, PAT$WRITE_MEM                              ;
                                   50      03  A8  9E 00473             MOVAB   3(R8), R0                                      ; 4007
                      7E           50          04  C7 00477             DIVL3   #4, R0, -(SP)                                  ;
                                               57  DD 0047B             PUSHL   NEW_INS_PTR                                    ;
                        00000000G  EF          02  FB 0047D             CALLS   #2, PAT$FREERELEASE                            ;
                                               0E  11 00484             BRB     34$                                            ; 4000
                                               58  DD 00486 33$:        PUSHL   R8                                             ; 4010
                                           F9  AD  9F 00488             PUSHAB  BR_INSTRUC+1                                   ;
                                               59  DD 0048B             PUSHL   R9                                             ;
                        00000000G  EF          03  FB 0048D             CALLS   #3, PAT$WRITE_MEM                              ;
                                   5A          59  D0 00494 34$:        MOVL    R9, NEXT_LOC                                   ; 4015
                                   59          58  C0 00497             ADDL2   R8, R9                                         ; 4016
                                   59          5A  D1 0049A 35$:        CMPL    NEXT_LOC, R9                                   ;
                                               20  18 0049D             BGEQ    36$                                            ;
                                               7E  D4 0049F             CLRL    -(SP)                                          ; 4019
                                   00000000G   EF  9F 004A1             PUSHAB  PAT$GL_NEW_ASD                                 ;
                                   00000000'   EF  9F 004A7             PUSHAB  NEW_TAB_STG                                    ;
                                               5A  DD 004AD             PUSHL   NEXT_LOC                                       ;
                        00000000V  EF          04  FB 004AF             CALLS   #4, PAT$OUT_MEM_LOC                            ;
                                   5A 00000000G  EF  D0 004B6           MOVL    PAT$GL_NEXT_LOC, NEXT_LOC                      ; 4020
                                               DB  11 004BD             BRB     35$                                            ; 4016
                                   5A          56  D0 004BF 36$:        MOVL    NEW_LOC, NEXT_LOC                              ; 4022
                                   50 00000000G  EF  D0 004C2 37$:      MOVL    PAT$GL_PATAREA, R0                             ; 4023
                             04    A0          5A  D1 004C9             CMPL    NEXT_LOC, 4(R0)                                ;
                                               20  18 004CD             BGEQ    38$                                            ;
                                               7E  D4 004CF             CLRL    -(SP)                                          ; 4026
                                   00000000G   EF  9F 004D1             PUSHAB  PAT$GL_NEW_ASD                                 ;
                                   00000000'   EF  9F 004D7             PUSHAB  NEW_TAB_STG                                    ;
                                               5A  DD 004DD             PUSHL   NEXT_LOC                                       ;
                        00000000V  EF          04  FB 004DF             CALLS   #4, PAT$OUT_MEM_LOC                            ;
                                   5A 00000000G  EF  D0 004E6           MOVL    PAT$GL_NEXT_LOC, NEXT_LOC                      ; 4027
                                               D3  11 004ED             BRB     37$                                            ; 4023
                                                   04 004EF 38$:        RET                                                    ; 4031

; Routine Size:  1264 bytes,    Routine Base:  _PAT$CODE + 066B
```

```
1 GLOBAL ROUTINE PAT$SUBST_INS (OLD_INS_PTR, INS_PC) =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 ! This routine substitutes other instruction sequences for branch-type
1 ! instructions that have been relocated to a new address and whose branch
1 ! displacements are now too small.  The following table describes the
1 ! possible substitutions.  If the branch in the first replacement choice does
1 ! not reach, then the second replacement choice is used.  Notice that the blank
1 ! lines in the table separate groups of instructions that are handled
1 ! similarly for substitutions.
1 !
1 ! OPC   INSTRUC        REPLACEMENT 1                     REPLACEMENT 2
1 ! ---   -------        -------------                     -------------
1 !
1 ! 12    BNEQ <X>       BEQL .+03, BRW <X>                BEQL .+06, JMP <X>
1 ! 13    BEQL <X>       BNEQ .+03, BRW <X>                BNEQ .+06, JMP <X>
1 ! 14    BGTR <X>       BLEQ .+03, BRW <X>                BLEQ .+06, JMP <X>
1 ! 15    BLEQ <X>       BGTR .+03, BRW <X>                BGTR .+06, JMP <X>
1 ! 18    BGEQ <X>       BLSS .+03, BRW <X>                BLSS .+06, JMP <X>
1 ! 19    BLSS <X>       BLSS .+03, BRW <X>                BLSS .+06, JMP <X>
1 ! 1A    BGTRU <X>      BLEQU .+03, BRW <X>               BLEQU .+06, JMP <X>
1 ! 1B    BLEQU <X>      BGTRU .+03, BRW <X>               BGTRU .+06, JMP <X>
1 ! 1C    BVC <X>        BVS .+03, BRW <X>                 BVS .+06, JMP <X>
1 ! 1D    BVS <X>        BVC .+03, BRW <X>                 BVC .+06, JMP <X>
1 ! 1E    BGEQU <X>      BLSSU .+03, BRW <X>               BLSSU .+06, JMP <X>
1 ! 1F    BLSSU <X>      BGEQU .+03, BRW <X>               BGEQU .+06, JMP <X>
1 ! E0    BBS <X>        BBC .+03, BRW <X>                 BBC .+06, JMP <X>
1 ! E1    BBC <X>        BBS .+03, BRW <X>                 BBS .+06, JMP <X>
1 ! E2    BBSS <X>       BBCS .+03, BRW <X>                BBCS .+06, JMP <X>
1 ! E3    BBCS <X>       BBSS .+06, BRW <X>                BBSS .+03, JMP <X>
1 ! E4    BBSC <X>       BBCC .+03, BRW <X>                BBCC .+06, JMP <X>
1 ! E5    BBCC <X>       BBSC .+03, BRW <X>                BBSC .+06, JMP <X>
1 ! E8    BLBS <X>       BLBC .+03, BRW <X>                BLBC .+06, JMP <X>
1 ! E9    BLBC <X>       BLBS .+03, BRW <X>                BLBS .+06, JMP <X>
1 !
1 ! E6    BBSSI <X>      BBSSI .+02, BRB .+03, BRW <X>     BBSSI .+02, BRB .+06, JMP <X>
1 ! E7    BBCCI <X>      BBCCI .+02, BRB .+03, BRW <X>     BBCCI .+02, BRB .+06, JMP <X>
1 ! F2    AOBLSS <X>     AOBLSS .+02, BRB .+03, BRW <X>    AOBLSS .+02, BRB .+06, JMP <X>
1 ! F3    AOBLEQ <X>     AOBLEQ .+02, BRB .+03, BRW <X>    AOBLEQ .+02, BRB .+06, JMP <X>
1 ! F4    SOBGEQ <X>     SOBGEQ .+02, BRB .+03, BRW <X>    SOBGEQ .+02, BRB .+06, JMP <X>
1 ! F5    SOBGTR <X>     SOBGTR .+02, BRB .+03, BRW <X>    SOBGTR .+02, BRB .+06, JMP <X>
1 !
1 ! 9D    ACBB <X>       ACBB .+02, BRB .+06, JMP <X>
1 ! 3D    ACBW <X>       ACBW .+02, BRB .+06, JMP <X>
1 ! F1    ACBL <X>       ACBL .+02, BRB .+06, JMP <X>
1 ! 4F    ACBF <X>       ACBF .+02, BRB .+06, JMP <X>
1 ! 6F    ACBD <X>       ACBD .+02, BRB .+06, JMP <X>
1 ! 4FFD  ACBG <X>       ACBG .+02, BRB .+06, JMP <X>
1 ! 6FFD  ACBH <X>       ACBH .+02, BRB .+06, JMP <X>
1 !
1 ! 11    BRB <X>        BRW <X>                           JMP <X>
1 ! 10    BSBB <X>       BSBW <X>                          JSB <X>
1 !
1 ! 31    BRW <X>        JMP <X>
1 ! 30    BSBW <X>       JSB <X>
```

```
1 !
1 ! In addition to the above instructions, there are also three case
1 ! instructions.  None of these has a replacement.  (In fact, the encoder
1 ! does not know how to insert a case instruction correctly; it will only
1 ! insert the instruction parameters.  The branch displacements must be
1 ! inserted as .WORD directives.)
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       PAT$SUBST_INS (OLD-ENCODED-INSTRUCTION-ADDRESS, PC-OF-INSTRUCTION)
1 !
1 ! INPUTS:
1 !
1 !       OLD_INS_PTR - Address of counted instruction stream to be substituted
1 !       INS_PC = Unmapped address of where to put instruction
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       PAT$GB_SUBST_IN - Buffer for substitution counted byte stream
1 !
1 ! OUTPUTS:
1 !
1 !       NONE
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       The substitution binary stream is written into INSTRUC_BUF
1 !       as a counted byte stream.
1 !
1 ! ROUTINE VALUE:
1 !
1 !       FALSE if no substitution instructions were possible.
1 !       TRUE if substitution was successful.
1 !
1 ! SIDE EFFECTS:
1 !
1 !       A substitution stream can now be written to memory, or an error
1 !       reported.  However, if an instruction had a label associated with it
1 !       any branches elsewhere in the code to it will no longer work!!!
1 !
1 !--
1
2 BEGIN
2
2 MAP
2       OLD_INS_PTR : REF VECTOR[,BYTE];                        ! Old binary instruction stream
2
2 LITERAL
2       MIN_WORD_DISP = -32768,                                 ! Minimum displacement for BRW
2       MAX_WORD_DISP = 32767,                                  ! Maximum displacement for BRW
2       BRB_OPCODE = %X'11',                                    ! Opcode for BRB instruction
2       BRW_OPCODE = %X'31',                                    ! Opcode for BRW instruction
2       JMP_OPCODE = %X'17',                                    ! Opcode forJMP instruction
2       BNEQ_OPCODE = %X'12',                                   ! Opcode for BNEQ instruction
2       BLEQ_OPCODE = %X'15',                                   ! Opcode for BLEQ instruction
2       BGEQ_OPCODE = %X'18',                                   ! Opcode for BGEQ instruction
2       BLSSU_OPCODE = %X'1F',                                  ! Opcode for BLSSU instruction
```

```
2               BBS_OPCODE = %X'E0',                                    ! Opcode for BBS instruction
2               BBCC_OPCODE = %X'E5',                                   ! Opcode for BBCC instruction
2               BLBS_OPCODE = %X'E8',                                   ! Opcode for BLBS instruction
2               BLBC_OPCODE = %X'E9',                                   ! Opcode for BLBC instruction
2               BBSSI_OPCODE = %X'E6',                                  ! Opcode for BBSSI instruction
2               BBCCI_OPCODE = %X'E7',                                  ! Opcode for BBCCI instruction
2               AOBLSS_OPCODE = %X'F2',                                 ! Opcode for AOBLSS instruction
2               SOBGTR_OPCODE = %X'F5',                                 ! Opcode for SOBGTR instruction
2               ACBB_OPCODE = %X'9D',                                   ! Opcode for ACBB instruction
2               ACBW_OPCODE = %X'3D',                                   ! Opcode for ACBW instruction
2               ACBL_OPCODE = %X'F1',                                   ! Opcode for ACBL instruction
2               ACBF_OPCODE = %X'4F',                                   ! Opcode for ACBF instruction
2               ACBD_OPCODE = %X'6F',                                   ! Opcode for ACBD instruction
2               ACBG_HICODE = %X'4F',                                   ! High byte of Opcode for ACBG instruction
2               ACBH_HICODE = %X'6F',                                   ! High byte of Opcode for ACBH instruction
2               CASEB_OPCODE = %X'8F',                                  ! Opcode for CASEB instruction
2               CASEW_OPCODE = %X'AF',                                  ! Opcode for CASEW instruction
2               CASEL_OPCODE = %X'CF',                                  ! Opcode for CASEL instruction
2               BSBW_OPCODE = %X'30',                                   ! Opcode for BSBW instruction
2               BSBB_OPCODE = %X'10',                                   ! Opcode for BSBB instruction
2               JSB_OPCODE = %X'16',                                    ! Opcode for JSB instruction
2               BRB_INS_SIZ = 2,                                        ! Size of BRB instruction
2               BRW_INS_SIZ = 3,                                        ! Size of BRW instruction
2               JMP_INS_SIZ = 6,                                        ! Size of JMP instruction
2               PC_DEFERRED = %X'EF',                                   ! PC deferred instruction mode
2               MAX_INST_LEN    = 80;                                   ! Maximum number of binary bytes in an instr
2
2       LOCAL
2               BR_DISPLACEMENT : SIGNED LONG;                          ! Displacement for branch instruction
2
2       !++
2       ! Handle the first group of substitutions.  These may be replaced with
2       ! their complement and a BRW, i.e., opcodes BGTR through BLBC in the above
2       ! table.  The complement instruction must be set to branch around the BRW
2       ! instruction.  Therefore, the instruction stream changes from:
2       !               <BR INS> TO <X>
2       ! TO:
2       !               <BR COM INS> TO .+03            BRW <X>
2       !--
2       IF (.OLD_INS_PTR[1] GEQU BNEQ_OPCODE AND .OLD_INS_PTR[1] LEQU BLEQ_OPCODE) OR
2          (.OLD_INS_PTR[1] GEQU BGEQ_OPCODE AND .OLD_INS_PTR[1] LEQU BLSSU_OPCODE) OR
2          (.OLD_INS_PTR[1] GEQU BBS_OPCODE  AND .OLD_INS_PTR[1] LEQU BBCC_OPCODE ) OR
3          (.OLD_INS_PTR[1] GEQU BLBS_OPCODE AND .OLD_INS_PTR[1] LEQU BLBC_OPCODE )
2       THEN
3               BEGIN
3               !++
3               ! Build the binary instruction stream for the complement branch.
3               ! Then build the BRW instruction with the old branch's displacement.
3               !--
3               PAT$GB_SUBST_IN[0] = BRW_INS_SIZ + .OLD_INS_PTR[0];     ! Set the entire stream length
3               PAT$GB_SUBST_IN[1] = (IF .OLD_INS_PTR[1] THEN (.OLD_INS_PTR[1] - 1)
3                                               ELSE (.OLD_INS_PTR[1] + 1));    ! Set complement opcode
3               CH$MOVE(.OLD_INS_PTR[0]-2, CH$PTR(OLD_INS_PTR[2]), CH$PTR(PAT$GB_SUBST_IN[2])); ! Move in instructio
3               PAT$GB_SUBST_IN[.OLD_INS_PTR[0]] = BRB_INS_SIZ;                ! Set complement branch around BRW instructi
3               PAT$GB_SUBST_IN[.OLD_INS_PTR[0]+1] = BRW_OPCODE;               ! Set BRW instruction opcode
3               BR_DISPLACEMENT = .PAT$GL_BR_DISPL + .OLD_INS_PTR[0] - .PAT$GB_SUBST_IN[0]; ! Compute new PC-relativ
3               IF (.BR_DISPLACEMENT LEQ MAX_WORD_DISP) AND
```

```
           (.BR_DISPLACEMENT GEQ MIN_WORD_DISP)                 ! Does displacement fit
        THEN
                (PAT$GB_SUBST_IN[ .OLD_INS_PTR[0]+2 ])<0,16,1> = .BR_DISPLACEMENT       ! Yes, move it into
        ELSE
                BEGIN
                !++
                ! No, it did not fit.  Use a JMP instead of a BRW, which
                ! is the second choice in the table.  The complement branch
                ! displacement must be changed, too.
                !--
                PAT$GB_SUBST_IN[0] = .PAT$GB_SUBST_IN[0] + (JMP_INS_SIZ - BRW_INS_SIZ); ! Set new instructio
                PAT$GB_SUBST_IN[ .OLD_INS_PTR[0] ] = JMP_INS_SIZ; ! Set complement branch around JMP instruc
                PAT$GB_SUBST_IN[ .OLD_INS_PTR[0]+1 ] = JMP_OPCODE; ! Set JMP opcode
                PAT$GB_SUBST_IN[ .OLD_INS_PTR[0]+2 ] = PC_DEFERRED; ! Set instruction mode
                (PAT$GB_SUBST_IN[ .OLD_INS_PTR[0]+3 ])<0,32,1> = .BR_DISPLACEMENT -
                                        (JMP_INS_SIZ - BRW_INS_SIZ); ! Set new branch displacement

                END
        END
ELSE
!++
! The opcode was not one of the first group, therefore check to see if
! it was one of the second group.
!--
IF (.OLD_INS_PTR[1] EQLU BBSSI_OPCODE) OR
   (.OLD_INS_PTR[1] EQLU BBCCI_OPCODE) OR
   (.OLD_INS_PTR[1] GEQU AOBLSS_OPCODE AND .OLD_INS_PTR[1] LEQU SOBGTR_OPCODE)
THEN
        BEGIN
        !++
        ! Handle the second group of substitutions.  THese may be replaced with
        ! the instruction branch, a BRB instruction, and a BRW or JMP
        ! instruction.  This group includes instructions BBSSI through ACBD in
        ! the above table.  The instruction branch must be set to branch around
        ! the BRB instruction.  The BRB instruction must be set to branch around
        ! the BRW instruction.  Therefore, the instruction stream changes from:
        !       <BR INS> TO <X>
        ! TO:
        !       <BR INS> TO .+02        BRB TO .+03     BRW <X>
        !--
        PAT$GB_SUBST_IN[0] = .OLD_INS_PTR[0] + BRB_INS_SIZ + BRW_INS_SIZ; ! Set the stream length
        CH$MOVE(.OLD_INS_PTR[0]-1, CH$PTR(OLD_INS_PTR[1]), CH$PTR(PAT$GB_SUBST_IN[1])); ! Copy old ins strea
        PAT$GB_SUBST_IN[ .OLD_INS_PTR[0] ] = BRB_INS_SIZ;               ! Set displ to br around BRB ins
        PAT$GB_SUBST_IN[ .OLD_INS_PTR[0]+1 ] = BRB_OPCODE;              ! Set BRB opcode
        PAT$GB_SUBST_IN[ .OLD_INS_PTR[0]+2 ] = BRW_INS_SIZ;             ! Set BRB around BRW ins
        PAT$GB_SUBST_IN[ .OLD_INS_PTR[0]+3 ] = BRW_OPCODE;              ! Set BRW opcode
        BR_DISPLACEMENT = .PAT$GL_BR_DISPL + .OLD_INS_PTR[0] - .PAT$GB_SUBST_IN[0]; ! Compute new PC-relativ
        IF (.BR_DISPLACEMENT LEQ MAX_WORD_DISP) AND
           (.BR_DISPLACEMENT GEQ MIN_WORD_DISP)                         ! Does displacement fit?
        THEN
                (PAT$GB_SUBST_IN[ .PAT$GB_SUBST_IN[0]-1 ])<0,16,1> = .BR_DISPLACEMENT ! Yes, move in displac
        ELSE
                BEGIN
                !++
                ! No, displacement did not fit, therefore use the
                ! second substitution choice.  This includes changing
                ! the BRW to a JMP, and altering the branch around it.
```

```
4                       !--
4                       PAT$GB_SUBST_IN[0] = .PAT$GB_SUBST_IN[0] + (JMP_INS_SIZ - BRW_INS_SIZ); ! Set a new stream s
4                       PAT$GB_SUBST_IN[ .OLD_INS_PTR[0]+2 ] = JMP_INS_SIZ; ! Change BRB displacement around JMP
4                       PAT$GB_SUBST_IN[ .OLD_INS_PTR[0]+3 ] = JMP_OPCODE; ! Replace the BRW opcode
4                       PAT$GB_SUBST_IN[ .OLD_INS_PTR[0]+4 ] = PC_DEFERRED; ! Set the instruction mode
4                       (PAT$GB_SUBST_IN[ .PAT$GB_SUBST_IN[0] + A_BYTE - A_LONGWORD ])<0,32,1> =
4                               .BR_DISPLACEMENT - (JMP_INS_SIZ - BRW_INS_SIZ); ! Adjust the displ
3                       END;
3               END
2       ELSE
2       !++
2       ! The opcode was not one of the second group, therefore check to see if it
2       ! was one of the third group.
2       !--
2       IF (.OLD_INS_PTR[1] EQLU ACBB_OPCODE) OR
2          (.OLD_INS_PTR[1] EQLU ACBW_OPCODE) OR
2          (.OLD_INS_PTR[1] EQLU ACBL_OPCODE) OR
2          (.OLD_INS_PTR[1] EQLU ACBF_OPCODE) OR
2          (.OLD_INS_PTR[1] EQLU ACBD_OPCODE) OR
2          (.OLD_INS_PTR[1] EQLU %X'FD' AND .OLD_INS_PTR[2] EQLU ACBG_HICODE) OR
3          (.OLD_INS_PTR[1] EQLU %X'FD' AND .OLD_INS_PTR[2] EQLU ACBH_HICODE)
2       THEN
3               BEGIN
3               !++
3               ! Handle the third group of substitutions.  These may be replaced with
3               ! the instruction branch, a BRB instruction, and a JMP instruction.
3               ! This group includes instructions ACBB through ACBD in the above table.
3               ! The instruction branch must be set to branch around the BRB
3               ! instruction.  The BRB instruction must be set to branch around the
3               ! BRW instruction.  THerefore, the instruction stream changes from:
3
3               ! CHANGES FROM:
3               !       <BR INS> TO <X>
3               ! TO:
3               !       <BR INS> TO .+02        BRB TO .+06     JMP <X>
3               !--
3               PAT$GB_SUBST_IN[0] = .OLD_INS_PTR[0] + (JMP_INS_SIZ + BRB_INS_SIZ); ! Set the stream length
3               CH$MOVE(.OLD_INS_PTR[0]-2, CH$PTR(OLD_INS_PTR[1]), CH$PTR(PAT$GB_SUBST_IN[1])); ! Copy old ins strea
3               PAT$GB_SUBST_IN[.OLD_INS_PTR[0]-1] = BRB_INS_SIZ;          ! Set displ to br around BRB ins
3               PAT$GB_SUBST_IN[.OLD_INS_PTR[0]] = 0;                      ! Clear other byte of displ word
3               PAT$GB_SUBST_IN[.OLD_INS_PTR[0]+1] = BRB_OPCODE;           ! Set BRB opcode
3               PAT$GB_SUBST_IN[.OLD_INS_PTR[0]+2] = JMP_INS_SIZ;          ! Set BRB around JMP instruction
3               PAT$GB_SUBST_IN[.OLD_INS_PTR[0]+3] = JMP_OPCODE;           ! Set BRW opcode
3               PAT$GB_SUBST_IN[.OLD_INS_PTR[0]+4] = PC_DEFERRED;          ! Set instruction mode
3               BR_DISPLACEMENT = .PAT$GL_BR_DISPL + .OLD_INS_PTR[0] - .PAT$GB_SUBST_IN[0]; ! Compute new PC-relativ
3               (PAT$GB_SUBST_IN[ .PAT$GB_SUBST_IN[0] + A_BYTE - A_LONGWORD ])<0,32,1> = .BR_DISPLACEMENT; ! Adjust
3               END
2       ELSE
2       !++
2       ! The opcode was not one of the third group, therefore check to see if it
2       ! was one of the fourth group.
2       !--
3       IF (.OLD_INS_PTR[1] EQL BRB_OPCODE) OR (.OLD_INS_PTR[1] EQL BSBB_OPCODE)
2       THEN
3               BEGIN
3               !++
3               ! Handle the fourth group of substitutions.  THese may be replaced with
```

```
3               ! the next larger displacement branch instruction of the same type.
3               ! This group includes instrucitons BRB and BSBB.  These instructions
3               ! can be handled similarly because:
3               ! (1) They have the same binary format, and
3               ! (2) The difference in opcodes for this branch
3               !     displacement and the next larger is the same.
3               ! Therefore, because of (1), the variables BRB_INS_SIZ and
3               ! BRW_INS_SIZ would be identical to BSBB_INS_SIZ and
3               ! BSBW_INS_SIZ.  Also, because of (2), (BRW_OPCODE - BRB_OPCODE)
3               ! is the same as (BSBW_OPCODE - BSBB_OPCODE).
3               !--
3               PAT$GB_SUBST_IN[0] = BRW_INS_SIZ;                       ! Set ins stream size
3               PAT$GB_SUBST_IN[1] = .OLD_INS_PTR[1] + (BRW_OPCODE - BRB_OPCODE); ! Set new opcode
3               BR_DISPLACEMENT = .PAT$GL_BR_DISPL + .OLD_INS_PTR[0] - .PAT$GB_SUBST_IN[0]; ! Compute new displ
3               IF (.BR_DISPLACEMENT LEQ MAX_WORD_DISP) AND
4                  (.BR_DISPLACEMENT GEQ MIN_WORD_DISP)                 ! Does displ fit?
3               THEN
3                       (PAT$GB_SUBST_IN[2])<0,16,1> = .BR_DISPLACEMENT ! Yes, move displ into stream
3               ELSE
4                       BEGIN
4                       !++
4                       ! No, displacement did not fit.  A longword displacement must be
4                       ! used.  Therefore, convert to a JSB or JMP instruction.
4                       !--
4                       PAT$GB_SUBST_IN[0] = JMP_INS_SIZ;               ! Set new stream size
4                       PAT$GB_SUBST_IN[1] = .PAT$GB_SUBST_IN[1] + (JMP_OPCODE - BRW_OPCODE); ! Set new opcode
4                       PAT$GB_SUBST_IN[2] = PC_DEFERRED;               ! Set instruction mode
4                       (PAT$GB_SUBST_IN[3])<0,32,1> = .BR_DISPLACEMENT -
4                               (JMP_INS_SIZ - BRW_INS_SIZ);            ! Compute new displacement
4                       END
3               END
2       ELSE
2       !++
2       ! The opcode was not one of the fourth group, therefore check to see if it
2       ! was one of the fifth group.
2       !--
3       IF (.OLD_INS_PTR[1] EQL BRW_OPCODE) OR (.OLD_INS_PTR[1] EQL BSBW_OPCODE)
2       THEN
3               BEGIN
3               !++
3               ! Handle the fifth group of substitutions.  These may be replaced with
3               ! the next larger displacement branch instruction of the same type.
3               ! This group includes instructions BRW and BSBW.  These instructions
3               ! can be handled similarly because:
3               ! (1) They have the same binary format, and
3               ! (2) The difference in opcodes for this branch displacement
3               !     and the next larger is the same.
3               ! Therefore, beacuse of (1), the variables JSB_INS_SIZ and JMP_INS_SIZ
3               ! would be identical.  Also, because of (2), (JMP_OPCODE - BRW_OPCODE)
3               ! is the same as (JSB_OPCODE - BSBW_OPCODE).
3               !--
3               PAT$GB_SUBST_IN[0] = JMP_INS_SIZ;                       ! Set ins stream size
3               PAT$GB_SUBST_IN[1] = .OLD_INS_PTR[1] + (JMP_OPCODE - BRW_OPCODE); ! Set opcode
3               PAT$GB_SUBST_IN[2] = PC_DEFERRED;                       ! Set ins mode
3               BR_DISPLACEMENT = .PAT$GL_BR_DISPL + .OLD_INS_PTR[0] - .PAT$GB_SUBST_IN[0]; ! Get displacement
3               (PAT$GB_SUBST_IN[3])<0,32,1> = .BR_DISPLACEMENT;        ! Compute br displ
3               END
```

```
; 1722        4374  2 ELSE
; 1723        4375  2         RETURN (FALSE);
; 1724        4376  2
; 1725        4377  2 RETURN (TRUE);
; 1726        4378  1 END;                                                    ! End of PAT$SUBST_INS



                                       0FFC 00000         .ENTRY  PAT$SUBST_INS, Save R2,R3,R4,R5,R6,R7,R8,-   ; 4032
                                                                  R9,R10,R11                                   ;
                     5B 00000000G   EF   9E 00002         MOVAB   PAT$GL_BR_DISPL, R11                         ;
                     5A 00000000G   EF   9E 00009         MOVAB   PAT$GB_SUBST_IN, R10                         ;
                     59        04   AC   D0 00010         MOVL    OLD_INS_PTR, R9                              ; 4185
                     57        01   A9   9A 00014         MOVZBL  1(R9), R7                                    ;
                     12             57   91 00018         CMPB    R7, #18                                      ;
                                    05   1F 0001B         BLSSU   1$                                           ;
                     15             57   91 0001D         CMPB    R7, #21                                      ;
                                    22   1B 00020         BLEQU   4$                                           ;
                     18             57   91 00022 1$:     CMPB    R7, #24                                      ; 4186
                                    05   1F 00025         BLSSU   2$                                           ;
                     1F             57   91 00027         CMPB    R7, #31                                      ;
                                    18   1B 0002A         BLEQU   4$                                           ;
                E0   8F             57   91 0002C 2$:     CMPB    R7, #224                                     ; 4187
                                    06   1F 00030         BLSSU   3$                                           ;
                E5   8F             57   91 00032         CMPB    R7, #229                                     ;
                                    0C   1B 00036         BLEQU   4$                                           ;
                E8   8F             57   91 00038 3$:     CMPB    R7, #232                                     ; 4188
                                    6B   1F 0003C         BLSSU   8$                                           ;
                E9   8F             57   91 0003E         CMPB    R7, #233                                     ;
                                    65   1A 00042         BGTRU   8$                                           ;
                     56             69   9A 00044 4$:     MOVZBL  (R9), R6                                     ; 4195
      6A             56             03   81 00047         ADDB3   #3, R6, PAT$GB_SUBST_IN                      ;
                     06             57   E9 0004B         BLBC    R7, 5$                                       ; 4196
                     50        FF   A7   9E 0004E         MOVAB   -1(R7), R0                                   ;
                                    04   11 00052         BRB     6$                                           ;
                     50        01   A7   9E 00054 5$:     MOVAB   1(R7), R0                                    ; 4197
                01   AA             50   90 00058 6$:     MOVB    R0, PAT$GB_SUBST_IN+1                        ; 4196
                     50        FE   A6   9E 0005C         MOVAB   -2(R6), R0                                   ; 4198
 02   AA        02   A9             50   28 00060         MOVC3   R0, 2(R9), PAT$GB_SUBST_IN+2                 ;
                   6A46             03   90 00066         MOVB    #3, PAT$GB_SUBST_IN[R6]                      ; 4199
                01 AA46             31   90 0006A         MOVB    #49, PAT$GB_SUBST_IN+1[R6]                   ; 4200
      50             6B             56   C1 0006F         ADDL3   R6, PAT$GL_BR_DISPL, R0                      ; 4201
                     58             6A   9A 00073         MOVZBL  PAT$GB_SUBST_IN, BR_DISPLACEMENT             ;
      58             50             58   C3 00076         SUBL3   BR_DISPLACEMENT, R0, BR_DISPLACEMENT         ;
         00007FFF    8F             58   D1 0007A         CMPL    BR_DISPLACEMENT, #32767                      ; 4202
                                    0F   14 00081         BGTR    7$                                           ;
         FFFF8000    8F             58   D1 00083         CMPL    BR_DISPLACEMENT, #-32768                     ; 4203
                                    06   19 0008A         BLSS    7$                                           ;
                             02 AA46     9F 0008C         PUSHAB  PAT$GB_SUBST_IN+2[R6]                        ; 4205
                                    77   11 00090         BRB     10$                                          ;
                     6A             03   80 00092 7$:     ADDB2   #3, PAT$GB_SUBST_IN                          ; 4213
                   6A46             06   90 00095         MOVB    #6, PAT$GB_SUBST_IN[R6]                      ; 4214
                01 AA46             17   90 00099         MOVB    #23, PAT$GB_SUBST_IN+1[R6]                   ; 4215
                02 AA46             11   8E 0009E         MNEGB   #17, PAT$GB_SUBST_IN+2[R6]                   ; 4216
                             03 AA46     9F 000A3         PUSHAB  PAT$GB_SUBST_IN+3[R6]                        ; 4217
```

```
PATEXA                                                 K 7
V04-000                                     16-Sep-1984 00:30:29     VAX-11 Bliss-32 V4.0-742                Page  51
                                            14-Sep-1984 12:52:32     DISK$VMSMASTER:[PATCH.SRC]PATEXA.B32;1    (8)

                                      7E 11 000A7          BRB     12$                                       ;
                  E6        8F        57 91 000A9 8$:      CMPB    R7, #230                                  ; 4227
                                      12 13 000AD          BEQL    9$                                        ;
                  E7        8F        57 91 000AF          CMPB    R7, #231                                  ; 4228
                                      0C 13 000B3          BEQL    9$                                        ;
                  F2        8F        57 91 000B5          CMPB    R7, #242                                  ; 4229
                                      72 1F 000B9          BLSSU   14$                                       ;
                  F5        8F        57 91 000BB          CMPB    R7, #245                                  ;
                                      6C 1A 000BF          BGTRU   14$                                       ;
                            56        69 9A 000C1 9$:      MOVZBL  (R9), R6                                  ; 4243
          6A                56        05 81 000C4          ADDB3   #5, R6, PAT$GB_SUBST_IN                   ;
                            50   FF   A6 9E 000C8          MOVAB   -1(R6), R0                                ; 4244
   01     AA        01      A9        50 28 000CC          MOVC3   R0, 1(R9), PAT$GB_SUBST_IN+1              ;
                          6A46        02 90 000D2          MOVB    #2, PAT$GB_SUBST_IN[R6]                   ; 4245
                       01 AA46        11 90 000D6          MOVB    #17, PAT$GB_SUBST_IN+1[R6]                ; 4246
                       02 AA46        03 90 000DB          MOVB    #3, PAT$GB_SUBST_IN+2[R6]                 ; 4247
                       03 AA46        31 90 000E0          MOVB    #49, PAT$GB_SUBST_IN+3[R6]                ; 4248
          50                6B        56 C1 000E5          ADDL3   R6, PAT$GL_BR_DISPL, R0                   ; 4249
                            58        6A 9A 000E9          MOVZBL  PAT$GB_SUBST_IN, BR_DISPLACEMENT          ;
          58                50        58 C3 000EC          SUBL3   BR_DISPLACEMENT, R0, BR_DISPLACEMENT      ;
             00007FFF       8F        58 D1 000F0          CMPL    BR_DISPLACEMENT, #32767                   ; 4250
                                      15 14 000F7          BGTR    11$                                       ;
             FFFF8000       8F        58 D1 000F9          CMPL    BR_DISPLACEMENT, #-32768                  ; 4251
                                      0C 19 00100          BLSS    11$                                       ;
                            50        6A 9A 00102          MOVZBL  PAT$GB_SUBST_IN, R0                       ; 4253
                                 FF AA40 9F 00105          PUSHAB  PAT$GB_SUBST_IN-1[R0]                     ;
                            9E        58 B0 00109 10$:     MOVW    BR_DISPLACEMENT, @(SP)+                   ;
                                      1D 11 0010C          BRB     13$                                       ;
                            6A        03 80 0010E 11$:     ADDB2   #3, PAT$GB_SUBST_IN                       ; 4261
                       02 AA46        06 90 00111          MOVB    #6, PAT$GB_SUBST_IN+2[R6]                 ; 4262
                       03 AA46        17 90 00116          MOVB    #23, PAT$GB_SUBST_IN+3[R6]                ; 4263
                       04 AA46        11 8E 0011B          MNEGB   #17, PAT$GB_SUBST_IN+4[R6]                ; 4264
                            50        6A 9A 00120          MOVZBL  PAT$GB_SUBST_IN, R0                       ; 4265
                                 FD AA40 9F 00123          PUSHAB  PAT$GB_SUBST_IN-3[R0]                     ; 4266
                            9E   FD   A8 9E 00127 12$:     MOVAB   -3(R8), @(SP)+                            ;
                                      79 11 0012B 13$:     BRB     17$                                       ; 4227
                  9D        8F        57 91 0012D 14$:     CMPB    R7, #157                                  ; 4274
                                      31 13 00131          BEQL    16$                                       ;
                            3D        57 91 00133          CMPB    R7, #61                                   ; 4275
                                      2C 13 00136          BEQL    16$                                       ;
                  F1        8F        57 91 00138          CMPB    R7, #241                                  ; 4276
                                      26 13 0013C          BEQL    16$                                       ;
                  4F        8F        57 91 0013E          CMPB    R7, #79                                   ; 4277
                                      20 13 00142          BEQL    16$                                       ;
                  6F        8F        57 91 00144          CMPB    R7, #111                                  ; 4278
                                      1A 13 00148          BEQL    16$                                       ;
                  FD        8F        57 91 0014A          CMPB    R7, #253                                  ; 4279
                                      07 12 0014E          BNEQ    15$                                       ;
                  4F        8F   02   A9 91 00150          CMPB    2(R9), #79                                ;
                                      0D 13 00155          BEQL    16$                                       ;
                  FD        8F        57 91 00157 15$:     CMPB    R7, #253                                  ; 4280
                                      4B 12 0015B          BNEQ    18$                                       ;
                  6F        8F   02   A9 91 0015D          CMPB    2(R9), #111                               ;
                                      44 12 00162          BNEQ    18$                                       ;
                            56        69 9A 00164 16$:     MOVZBL  (R9), R6                                  ; 4296
          6A                56        08 81 00167          ADDB3   #8, R6, PAT$GB_SUBST_IN                   ;
                            50   FE   A6 9E 0016B          MOVAB   -2(R6), R0                                ; 4297
```

```
                                                        L 7
PATEXA                                             16-Sep-1984 00:30:29    VAX-11 Bliss-32 V4.0-742              Page  52
V04-000                                            14-Sep-1984 12:52:32    DISK$VMSMASTER:[PATCH.SRC]PATEXA.B32;1     (8)

          01   AA        01   A9            50  28 0016F          MOVC3   R0, 1(R9), PAT$GB_SUBST_IN+1                   :
                         FF AA46            02  90 00175          MOVB    #2, PAT$GB_SUBST_IN-1[R6]                      : 4298
                              6A46          94     0017A          CLRB    PAT$GB_SUBST_IN[R6]                            : 4299
                         01 AA46            11  90 0017D          MOVB    #17, PAT$GB_SUBST_IN+1[R6]                     : 4300
                         02 AA46            06  90 00182          MOVB    #6, PAT$GB_SUBST_IN+2[R6]                      : 4301
                         03 AA46            17  90 00187          MOVB    #23, PAT$GB_SUBST_IN+3[R6]                     : 4302
                         04 AA46            11  8E 0018C          MNEGB   #17, PAT$GB_SUBST_IN+4[R6]                     : 4303
          50                  6B            56  C1 00191          ADDL3   R6, PAT$GL_BR_DISPL, R0                        : 4304
                              58            6A  9A 00195          MOVZBL  PAT$GB_SUBST_IN, BR_DISPLACEMENT               :
          58                  50            58  C3 00198          SUBL3   BR_DISPLACEMENT, R0, BR_DISPLACEMENT           :
                              50            6A  9A 0019C          MOVZBL  PAT$GB_SUBST_IN, R0                            : 4305
                                      FD AA40   9F 0019F          PUSHAB  PAT$GB_SUBST_IN-3[R0]                          :
                              9E            58  D0 001A3          MOVL    BR_DISPLACEMENT, @(SP)+                        :
                                            70  11 001A6 17$:     BRB     23$                                            : 4274
                              11            57  91 001A8 18$:     CMPB    R7, #17                                        : 4312
                                            05  13 001AB          BEQL    19$                                            :
                              10            57  91 001AD          CMPB    R7, #16                                        :
                                            3F  12 001B0          BNEQ    21$                                            :
                              6A            03  90 001B2 19$:     MOVB    #3, PAT$GB_SUBST_IN                            : 4328
          01   AA             57            20  81 001B5          ADDB3   #32, R7, PAT$GB_SUBST_IN+1                     : 4329
                              50            69  9A 001BA          MOVZBL  (R9), R0                                       : 4330
                              50            6B  C0 001BD          ADDL2   PAT$GL_BR_DISPL, R0                            :
                              58            6A  9A 001C0          MOVZBL  PAT$GB_SUBST_IN, BR_DISPLACEMENT               :
          58                  50            58  C3 001C3          SUBL3   BR_DISPLACEMENT, R0, BR_DISPLACEMENT           :
                00007FFF      8F            58  D1 001C7          CMPL    BR_DISPLACEMENT, #32767                        : 4331
                                            0F  14 001CE          BGTR    20$                                            :
                FFFF8000      8F            58  D1 001D0          CMPL    BR_DISPLACEMENT, #-32768                       : 4332
                                            06  19 001D7          BLSS    20$                                            :
                         02   AA            58  B0 001D9          MOVW    BR_DISPLACEMENT, PAT$GB_SUBST_IN+2             : 4334
                                            39  11 001DD          BRB     23$                                            :
                              6A            06  90 001DF 20$:     MOVB    #6, PAT$GB_SUBST_IN                            : 4341
                         01   AA            1A  82 001E2          SUBB2   #26, PAT$GB_SUBST_IN+1                         : 4342
                         02   AA            11  8E 001E6          MNEGB   #17, PAT$GB_SUBST_IN+2                         : 4343
                         03   AA      FD    A8  9E 001EA          MOVAB   -3(R8), PAT$GB_SUBST_IN+3                      : 4344
                                            27  11 001EF          BRB     23$                                            : 4331
                              31            57  91 001F1 21$:     CMPB    R7, #49                                        : 4353
                                            05  13 001F4          BEQL    22$                                            :
                              30            57  91 001F6          CMPB    R7, #48                                        :
                                            21  12 001F9          BNEQ    24$                                            :
                              6A            06  90 001FB 22$:     MOVB    #6, PAT$GB_SUBST_IN                            : 4368
          01   AA             57            1A  83 001FE          SUBB3   #26, R7, PAT$GB_SUBST_IN+1                     : 4369
                         02   AA            11  8E 00203          MNEGB   #17, PAT$GB_SUBST_IN+2                         : 4370
                              50            69  9A 00207          MOVZBL  (R9), R0                                       : 4371
                              50            6B  C0 0020A          ADDL2   PAT$GL_BR_DISPL, R0                            :
                              58            6A  9A 0020D          MOVZBL  PAT$GB_SUBST_IN, BR_DISPLACEMENT               :
          58                  50            58  C3 00210          SUBL3   BR_DISPLACEMENT, R0, BR_DISPLACEMENT           :
                         03   AA            58  D0 00214          MOVL    BR_DISPLACEMENT, PAT$GB_SUBST_IN+3             : 4372
                              50            01  D0 00218 23$:     MOVL    #1, R0                                         : 4377
                                                04 0021B          RET                                                    :
                                            50  D4 0021C 24$:     CLRL    R0                                             : 4378
                                                04 0021E          RET                                                    :

; Routine Size:  543 bytes,    Routine Base:  _PAT$CODE + 0B5B
```

```
1 GLOBAL ROUTINE PAT$OUT_MEM_LOC (LOCATION, PREFIX_STG, ASM_DIR_TBL, CASE_TBL) =
1
1 !++
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Outputs the value of a memory location to the output
1 !       device. If this routine is called as a result of an EXAMINE
1 !       command, the location itself is also displayed, followed by
1 !       a colon and a tab.
1 !
1 !       The appropriate mode settings are used to control the output
1 !       style.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       PAT$OUT_MEM_LOC ()
1 !
1 ! INPUTS:
1 !
1 !       LOCATION        - Unmapped location whose contents are to be displayed.
1 !       PREFIX_STG      - Prefix string to output before the location
1 !                         0 = NONE
1 !       ASM_DIR_TBL     - Address of assembler directive table descriptor
1 !       CASE_TBL        - TRUE => Print CASE dispatch tables
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       PAT$GL_CONTEXT [EXAMINE_BIT]    - If this bit is set, the address of the
1 !                                         value is also displayed.
1 !       PAT$GL_MOD_PTR                  - Pointer to the current mode level
1 !
1 ! OUTPUTS:
1 !
1 !       TRUE for success, FALSE for failure.
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       NONE
1 !
1 ! ROUTINE VALUE:
1 !
1 !       TRUE or FALSE
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Data is output to the data device.  An error message is produced if the
1 !       memory location is not readable.
1 !
1 !--
1
2 BEGIN
2 LOCAL
2       MAPPED_LOC : REF VECTOR[,BYTE],                         ! Mapped address of deposit location
2       ISE_ADDR: REF VECTOR[,BYTE],                            ! ISE address for deposit location
2       OUT_VALUES : VECTOR[ TTY_OUT_WIDTH, BYTE],
2       OUTPUT_BUFFER : VECTOR[ TTY_OUT_WIDTH, BYTE];
2
```

```
2  !++
2  ! Initialize buffer address and size.
2  !--
2  PAT$CP_OUT_STR = OUTPUT_BUFFER + 1;
2  PAT$GL_BUF_SIZ = 0;
2
2  !++
2  ! First check if there is a prefix string to be output.
2  !--
3  IF (.PREFIX_STG NEQ 0)
2  THEN
2          PAT$FAO_PUT(.PREFIX_STG);
2
2  !++
2  ! Now if the examine bit is set then output a location which is mapped
2  ! by PATCH.  If the examine bit is not set, then output an expression
2  ! for the EVALUATE command.
2  !--
2  IF .PAT$GL_CONTEXT [EXAMINE_BIT]
2  THEN
3          BEGIN
3          !++
3          ! Print the address, making it come out as a longword regardless of
3          ! the current output mode length.
3          !--
3          PAT$MAP_ADDR(.LOCATION,MAPPED_LOC,ISE_ADDR);              ! Compute mapped address
3          PAT$OUT_SYM_VAL(.LOCATION, LONG_LENGTH, 0);
3          PAT$GL_LAST_LOC = .LOCATION;
3          PAT$GB_LOC_TYPE = MEMORY_LOC;
3          PAT$FAO_PUT ( COLON_TAB_STG );
3
3          !++
3          ! Handle output as symbolic instructions separately.
3          !--
4          IF( .PAT$GB_MOD_PTR[ MODE_INSTRUC ] )
3          THEN
4                  IF ((LOCATION = PAT$INS_DECODE (.LOCATION, OUTPUT_BUFFER, LOCATION, .ASM_DIR_TBL, .CASE_TBL)
3                  THEN
4                          BEGIN
4                          SIGNAL (PAT$_NODECODE);
4                          RETURN(FALSE);
4                          END
3                  ELSE
4                          BEGIN
4                          PAT$MAP_ADDR (.LOCATION, MAPPED_LOC, ISE_ADDR);
4                          IF .PAT$GL_CONTEXT [EXAMINE_BIT]
4                          THEN
4                                  PAT$GL_NEXT_LOC = .LOCATION;
4
4                          !++
4                          ! PAT$GL_LAST_VAL may be set within PAT$INS_DECODE.
4                          !--
4                          END
3          ELSE
4                  BEGIN
4                  !++
4                  ! Special attention for ascii output.
```

```
4                   !--
5                   IF (.PAT$GB_MOD_PTR [MODE_ASCII])
4                   THEN
4                           !++
4                           ! Simply output the number of characters
4                           ! implied by the current MODE_LENGTH setting.
4                           !--
5                           BEGIN
5                           PAT$GET_VALUE (.LOCATION, .PAT$GB_MOD_PTR[MODE_LENGTH], OUT_VALUES);
5                           PAT$FAO_PUT (CS_ASCII, .PAT$GB_MOD_PTR[MODE_LENGTH], OUT_VALUES);
5                           PAT$GL_NEXT_LOC = .LOCATION + .PAT$GB_MOD_PTR [MODE_LENGTH];
5                           PAT$GL_LAST_VAL = .(.MAPPED_LOC) <0, .PAT$GB_MOD_PTR [MODE_LENGTH] * 8>;
5                           END
4                   ELSE                                            ! Otherwise we have one of the usual modes
4                           IF .PAT$GL_CONTEXT [EXAMINE_BIT]
4                           THEN
5                                   BEGIN
5                                   PAT$GET_VALUE(.LOCATION, .PAT$GB_MOD_PTR[MODE_LENGTH], OUT_VALUES);
5                                   PAT$OUT_NUM_VAL(.OUT_VALUES, 0, 0, TRUE);
5                                   PAT$GL_NEXT_LOC = .LOCATION + .PAT$GB_MOD_PTR [MODE_LENGTH];
5                                   PAT$GL_LAST_VAL = .OUT_VALUES <0, .PAT$GB_MOD_PTR [MODE_LENGTH] * 8>;
4                                   END;
4                   END
3           END
2   ELSE
3           BEGIN
3           !++
3           ! Output the value for the EVALUATE command here then return.
3           ! All other commands have set the examine bit.  Check for different
3           ! output modes, literal or instruction.
3           !--
4           IF (.PAT$GL_CONTEXT[LITERAL_BIT])
3           THEN
4                   BEGIN
4                   !++
4                   ! Call a routine which does the whole thing - including
4                   ! flushing the output and producing an error message if no
4                   ! such literal translation can be found.
4                   !--
4                   DISPLAY_LVTS(..LOCATION);
4
4                   !++
4                   ! If the above routine returns then at least one literal
4                   ! translation was found.  This form of evaluate sets the
4                   ! psuedo '\' (last value displayed) only.
4                   !--
4                   PAT$GL_LAST_VAL = ..LOCATION;
4                   RETURN(TRUE);
3                   END;
3
3           !++
3           ! Instruction mode works only if /LITERAL was not specified.
3           !--
4           IF (.PAT$GB_MOD_PTR[MODE_INSTRUC])
3           THEN
4                   BEGIN
4                   LOCAL
```

```
; 1899        4550  4                   COUNT,
; 1900        4551  4                   ENCODED_BUF : VECTOR[38,BYTE];
; 1901        4552  5           IF (NOT PAT$INS_ENCODE(..LOCATION, ENCODED_BUF, 0))
; 1902        4553  4           THEN
; 1903        4554  4                   SIGNAL(PAT$_NOENCODE, 1, ..LOCATION);
; 1904        4555  4           COUNT = .ENCODED_BUF[0];
; 1905        4556  4           DO
; 1906        4557  5                   BEGIN
; 1907        4558  5                   PAT$OUT_NUM_VAL(.ENCODED_BUF[.COUNT], BYTE_LENGTH, HEX_RADIX, FALSE);
; 1908        4559  5                   COUNT = .COUNT - 1;
; 1909        4560  5                   END
; 1910        4561  4           UNTIL .COUNT EQL 0;
; 1911        4562  4           END
; 1912        4563  3       ELSE
; 1913        4564  4           BEGIN
; 1914        4565  4           PAT$OUT_NUM_VAL(..LOCATION, 0, 0, TRUE);
; 1915        4566  4           PAT$GL_LAST_VAL = .(LOCATION) <0, .PAT$GB_MOD_PTR [MODE_LENGTH] * 8>;
; 1916        4567  3           END;
; 1917        4568  2       END;
; 1918        4569  2
; 1919        4570  2 !++
; 1920        4571  2 ! Write out the string and return.
; 1921        4572  2 !--
; 1922        4573  2 PAT$OUT_PUT( OUTPUT_BUFFER );
; 1923        4574  2
; 1924        4575  2 RETURN TRUE
; 1925        4576  1 END;



                                   0FFC 00000          .ENTRY  PAT$OUT_MEM_LOC, Save R2,R3,R4,R5,R6,R7,R8,-;  4379
                                                               R9,R10,R11                                   ;
                58 00000000G  00  9E 00002          MOVAB   LIB$SIGNAL, R11                              ;
                5A 00000000G  EF  9E 00009          MOVAB   PAT$MAP_ADDR, R10                            ;
                59 00000000G  EF  9E 00010          MOVAB   PAT$OUT_NUM_VAL, R9                          ;
                58 00000000G  EF  9E 00017          MOVAB   PAT$GL_NEXT_LOC, R8                          ;
                57 00000000G  EF  9E 0001E          MOVAB   PAT$FAO_PUT, R7                              ;
                56 00000000G  EF  9E 00025          MOVAB   PAT$GL_LAST_VAL, R6                          ;
                55 00000000G  EF  9E 0002C          MOVAB   PAT$GL_CONTEXT, R5                           ;
                54 00000000G  EF  9E 00033          MOVAB   PAT$GB_MOD_PTR, R4                           ;
                5E      FEC8  CE  9E 0003A          MOVAB   -312(SP), SP                                 ;
     00000000G  EF        31  AE  9E 0003F          MOVAB   OUTPUT_BUFFER+1, PAT$CP_OUT_STR              ;  4439
                   00000000G  EF  D4 00047          CLRL    PAT$GL_BUF_SIZ                               ;  4440
                          08  AC  D5 0004D          TSTL    PREFIX_STG                                   ;  4445
                              06  13 00050          BEQL    1$                                           ;
                          08  AC  DD 00052          PUSHL   PREFIX_STG                                   ;  4447
                67            01  FB 00055          CALLS   #1, PAT$FAO_PUT                              ;
                53        04  AC  D0 00058 1$:      MOVL    LOCATION, R3                                 ;  4461
                03        01  A5  E8 0005C          BLBS    PAT$GL_CONTEXT+1, 2$                         ;  4454
                            00EB  31 00060          BRW     7$                                           ;
                              5E  DD 00063 2$:      PUSHL   SP                                           ;  4461
                          08  AE  9F 00065          PUSHAB  MAPPED_LOC                                   ;
                              53  DD 00068          PUSHL   R3                                           ;
                6A            03  FB 0006A          CALLS   #3, PAT$MAP_ADDR                             ;
                7E            04  7D 0006D          MOVQ    #4, -(SP)                                    ;  4462
```

```
PATEXA                                                            D  8
V04-000                                                  16-Sep-1984 00:30:29    VAX-11 Bliss-32 V4.0-742              Page  57
                                                         14-Sep-1984 12:52:32    DISK$VMSMASTER:[PATCH.SRC]PATEXA.B32;1  (9)

                                            53  DD 00070          PUSHL    R3                                        :
                   00000000G  EF            03  FB 00072          CALLS    #3, PAT$OUT_SYM_VAL                       :
                   00000000G  EF            53  D0 00079          MOVL     R3, PAT$GL_LAST_LOC                       : 4463
                         00000000G  EF      94  00080             CLRB     PAT$GB_LOC_TYPE                           : 4464
                         00000000'  EF      9F  00086             PUSHAB   COLON_TAB_STG                             : 4465
                              67            01  FB 0008C          CALLS    #1, PAT$FAO_PUT                           :
                              52            64  D0 0008F          MOVL     PAT$GB_MOD_PTR, R2                        : 4470
                              3A        03  A2  E9 00092          BLBC     3(R2), 4$                                 :
                              7E        0C  AC  7D 00096          MOVQ     ASM_DIR_TBL, -(SP)                        : 4472
                                        04  AC  9F 0009A          PUSHAB   LOCATION                                  :
                                        3C  AE  9F 0009D          PUSHAB   OUTPUT_BUFFER                             :
                                            53  DD 000A0          PUSHL    R3                                        :
                   00000000G  EF            05  FB 000A2          CALLS    #5, PAT$INS_DECODE                        :
                          04  AC            50  D0 000A9          MOVL     R0, LOCATION                              :
                                            0C  12 000AD          BNEQ     3$                                        :
                         006D8102  8F           DD 000AF          PUSHL    #7176450                                  : 4475
                              6B            01  FB 000B5          CALLS    #1, LIB$SIGNAL                            :
                                        010B    31 000B8          BRW      14$                                       : 4476
                                            5E  DD 000BB  3$:     PUSHL    SP                                        : 4480
                                        08  AE  9F 000BD          PUSHAB   MAPPED_LOC                                :
                                        04  AC  DD 000C0          PUSHL    LOCATION                                  :
                              6A            03  FB 000C3          CALLS    #3, PAT$MAP_ADDR                          :
                              49        01  A5  E9 000C6          BLBC     PAT$GL_CONTEXT+1, 5$                      : 4481
                              68        04  AC  D0 000CA          MOVL     LOCATION, PAT$GL_NEXT_LOC                 : 4483
                                            7C  11 000CE          BRB      6$                                        : 4472
                              3F        04  A2  E9 000D0  4$:     BLBC     4(R2), 5$                                 : 4494
                                      FF7C  CD  9F 000D4          PUSHAB   OUT_VALUES                                : 4501
                              7E        01  A2  9A 000D8          MOVZBL   1(R2), -(SP)                              :
                                            53  DD 000DC          PUSHL    R3                                        :
                   00000000G  EF            03  FB 000DE          CALLS    #3, PAT$GET_VALUE                         :
                                      FF7C  CD  9F 000E5          PUSHAB   OUT_VALUES                                : 4502
                              50            64  D0 000E9          MOVL     PAT$GB_MOD_PTR, R0                        :
                              7E        01  A0  9A 000EC          MOVZBL   1(R0), -(SP)                              :
                         00000000'  EF      9F  000F0             PUSHAB   CS_ASCII                                  :
                              67            03  FB 000F6          CALLS    #3, PAT$FAO_PUT                           :
                              50            64  D0 000F9          MOVL     PAT$GB_MOD_PTR, R0                        : 4503
                              51        01  A0  9A 000FC          MOVZBL   1(R0), R1                                 :
                        68    53            51  C1 00100          ADDL3    R1, R3, PAT$GL_NEXT_LOC                   :
                              50        01  A0  9A 00104          MOVZBL   1(R0), R0                                 : 4504
                              50            08  C4 00108          MULL2    #8, R0                                    :
         66       04    BE    50            00  EF 0010B          EXTZV    #0, R0, @MAPPED_LOC, PAT$GL_LAST_VAL      :
                                            39  11 00111          BRB      6$                                        : 4494
                              35        01  A5  E9 00113  5$:     BLBC     PAT$GL_CONTEXT+1, 6$                      : 4507
                                      FF7C  CD  9F 00117          PUSHAB   OUT_VALUES                                : 4510
                              7E        01  A2  9A 0011B          MOVZBL   1(R2), -(SP)                              :
                                            53  DD 0011F          PUSHL    R3                                        :
                   00000000G  EF            03  FB 00121          CALLS    #3, PAT$GET_VALUE                         :
                                            01  DD 00128          PUSHL    #1                                        : 4511
                                            7E  7C 0012A          CLRQ     -(SP)                                     :
                                      FF7C  CD  DD 0012C          PUSHL    OUT_VALUES                                :
                              69            04  FB 00130          CALLS    #4, PAT$OUT_NUM_VAL                       :
                              50            64  D0 00133          MOVL     PAT$GB_MOD_PTR, R0                        : 4512
                              51        01  A0  9A 00136          MOVZBL   1(R0), R1                                 :
                        68    53            51  C1 0013A          ADDL3    R1, R3, PAT$GL_NEXT_LOC                   :
                              50        01  A0  9A 0013E          MOVZBL   1(R0), R0                                 : 4513
                              50            08  C4 00142          MULL2    #8, R0                                    :
         66     FF7C    CD    50            00  EF 00145          EXTZV    #0, R0, OUT_VALUES, PAT$GL_LAST_VAL       :
```

```
                                      6A  11 0014C 6$:     BRB     12$                                      ; 4470
           10      03   A5            01  E1 0014E 7$:     BBC     #1, PAT$GL_CONTEXT+3, 8$                 ; 4524
                                 04   BC  DD 00153         PUSHL   @LOCATION                                ; 4532
              00000000V   EF          01  FB 00156         CALLS   #1, DISPLAY_LVTS                         ;
                          66     04   BC  D0 0015D         MOVL    @LOCATION, PAT$GL_LAST_VAL               ; 4539
                                      5F  11 00161         BRB     13$                                      ; 4540
                          50          64  D0 00163 8$:     MOVL    PAT$GB_MOD_PTR, R0                       ; 4546
                          35     03   A0  E9 00166         BLBC    3(R0), 11$                               ;
                                      7E  D4 0016A         CLRL    -(SP)                                    ; 4552
                                 0C   AE  9F 0016C         PUSHAB  ENCODED_BUF                              ;
                                      63  DD 0016F         PUSHL   (R3)                                     ;
              00000000G   EF          03  FB 00171         CALLS   #3, PAT$INS_ENCODE                       ;
                          0D          50  E8 00178         BLBS    R0, 9$                                   ;
                                      63  DD 0017B         PUSHL   (R3)                                     ; 4554
                                      01  DD 0017D         PUSHL   #1                                       ;
                       006D810A       8F  DD 0017F         PUSHL   #7176458                                 ;
                       6B             03  FB 00185         CALLS   #3, LIB$SIGNAL                           ;
                       52        08   AE  9A 00188 9$:     MOVZBL  ENCODED_BUF, COUNT                       ; 4555
                       7E             10  7D 0018C 10$:    MOVQ    #16, -(SP)                               ; 4558
                                      01  DD 0018F         PUSHL   #1                                       ;
                       7E        14 AE42  9A 00191         MOVZBL  ENCODED_BUF[COUNT], -(SP)                ;
                       69             04  FB 00196         CALLS   #4, PAT$OUT_NUM_VAL                      ;
                                      52  D7 00199         DECL    COUNT                                    ; 4559
                                      EF  12 0019B         BNEQ    10$                                      ; 4561
                                      19  11 0019D         BRB     12$                                      ; 4546
                                      01  DD 0019F 11$:    PUSHL   #1                                       ; 4565
                                      7E  7C 001A1         CLRQ    -(SP)                                    ;
                                      63  DD 001A3         PUSHL   (R3)                                     ;
                       69             04  FB 001A5         CALLS   #4, PAT$OUT_NUM_VAL                      ;
                       50             64  D0 001A8         MOVL    PAT$GB_MOD_PTR, R0                       ; 4566
                       50        01   A0  9A 001AB         MOVZBL  1(R0), R0                                ;
                       50             08  C4 001AF         MULL2   #8, R0                                   ;
    66       04   AC   50             00  EF 001B2         EXTZV   #0, R0, LOCATION, PAT$GL_LAST_VAL        ;
                                 30   AE  9F 001B8 12$:    PUSHAB  OUTPUT_BUFFER                            ; 4573
              00000000G   EF          01  FB 001BB         CALLS   #1, PAT$OUT_PUT                          ;
                          50          01  D0 001C2 13$:    MOVL    #1, R0                                   ; 4575
                                          04 001C5         RET                                              ;
                                      50  D4 001C6 14$:    CLRL    R0                                       ; 4576
                                          04 001C8         RET                                              ;

; Routine Size:  457 bytes,    Routine Base:  _PAT$CODE + 0D7A
```

```
1 ROUTINE DISPLAY_LVTS (VALUE) : NOVALUE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Given a value, display the pathnames of all literals in the LVT which
1 !       have this value.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       CALLS #1, DISPLAY_LVTS
1 !
1 ! INPUTS:
1 !
1 !       VALUE - Literal value to be translated to symbols
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       The initial set up for standard PATCH I/O has already been done.
1 !       This routine (re)uses this buffer for its output.
1 !
1 ! OUTPUTS:
1 !
1 !       none
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       All the literal symbols associated with the value are printed.
1 !
1 ! ROUTINE VALUE:
1 !
1 !       novalue
1 !
1 ! SIDE EFFECTS:
1 !
1 !       Either output is sent to SYS$OUTPUT or a SIGNAL is generated and
1 !       no return is done.
1 !
1 !--
1
2 BEGIN
2
2 LOCAL
2       OUTPUT_BUFFER : REF VECTOR[,BYTE],                      ! Output buffer for SYS$OUTPUT writes
2       LVT_PTR : REF LVT_RECORD,                               ! Pointer to LVT match
2       ONE_FOUND;                                              ! Indicator if at least one symbol was found
2
2 !++
2 ! Initialize a flag which is used to know whether or not at least one match
2 ! to the given value has been found.  Also save a pointer to current output
2 ! buffer so that it can be reused.
2 !--
2 ONE_FOUND = FALSE;
2 OUTPUT_BUFFER = .PAT$CP_OUT_STR;
2
2 !++
; 1984          4634  2 ! Access to the LVT is via a 'canned' function.  Before using it, this routine
; 1985          4635  2 ! must signal its intention to do so.
; 1986          4636  2 !--
; 1987          4637  2 PAT$GET_NXT_LVT(SL_ACCE_INIT);
; 1988          4638  2
; 1989          4639  2 !++
; 1990          4640  2 ! Loop through the LVT sequentially, asking to see all currently valid records.
; 1991          4641  2 !--
; 1992          4642  3 WHILE ((LVT_PTR = PAT$GET_NXT_LVT(SL_ACCE_RECS)) NEQA 0)
; 1993          4643  3         DO
; 1994          4644  3         BEGIN
; 1995          4645  4         IF (.LVT_PTR[LVT_VALUE] EQL .VALUE)
; 1996          4646  3         THEN
; 1997          4647  4                 BEGIN
; 1998          4648  4                 LOCAL
; 1999          4649  4                         NT_PTR : REF NT_RECORD,
; 2000          4650  4                         PATH_VEC : PATHNAME_VECTOR;
; 2001          4651  4
; 2002          4652  4                 !++
; 2003          4653  4                 ! Found a match.  Print out the corresponding pathname by
; 2004          4654  4                 ! first building a pathname vector based on the returned NT_PTR.
; 2005          4655  4                 !--
; 2006          4656  4                 ONE_FOUND = TRUE;
; 2007          4657  4                 NT_PTR = .LVT_PTR[LVT_NT_PTR];
; 2008          4658  4                 PAT$ADD_NT_T_PV(.NT_PTR, PATH_VEC);
; 2009          4659  4                 PAT$PRINT_PATH(PATH_VEC);
; 2010          4660  4
; 2011          4661  4                 !++
; 2012          4662  4                 ! Write out the string and reset the global buffer pointers.
; 2013          4663  4                 !--
; 2014          4664  4                 PAT$OUT_PUT(.OUTPUT_BUFFER-1);
; 2015          4665  4                 PAT$CP_OUT_STR = .OUTPUT_BUFFER;
; 2016          4666  4                 PAT$GL_BUF_SIZ = 0;
; 2017          4667  3                 END;
; 2018          4668  2         END;                                            ! Loop back to consider the next LVT record
; 2019          4669  2
; 2020          4670  2 !++
; 2021          4671  2 ! At this point, the LVT has been completely searched.  If no matches were
; 2022          4672  2 ! found, then signal a warning.
; 2023          4673  2 !--
; 2024          4674  3 IF (NOT .ONE_FOUND)
; 2025          4675  2 THEN
; 2026          4676  2         SIGNAL(PAT$_NOLITERAL+MSG$K_WARN, 1, .VALUE);
; 2027          4677  2 RETURN;
; 2028          4678  1 END;                                                    ! End of DISPLAY_LVTS



                                    007C 00000 DISPLAY_LVTS:
                                                     .WORD    Save R2,R3,R4,R5,R6                     ; 4577
                  56 00000000G   EF   9E 00002       MOVAB    PAT$CP_OUT_STR, R6                      ;
                  55 00000000G   EF   9E 00009       MOVAB    PAT$GET_NXT_LVT, R5                     ;
                  5E             2C   C2 00010       SUBL2    #44, SP                                 ;
                                 54   D4 00013       CLRL     ONE_FOUND                               ; 4630
                  53             66   D0 00015       MOVL     PAT$CP_OUT_STR, OUTPUT_BUFFER           ; 4631
```

```
                                                                     H  8
PATEXA                                                    16-Sep-1984 00:30:29    VAX-11 Bliss-32 V4.0-742                  Page  61
V04-000                                                   14-Sep-1984 12:52:32    DISK$VMSMASTER:[PATCH.SRC]PATEXA.B32;1    (10)

                                      7E  D4 00018          CLRL    -(SP)                                                   ; 4637
                    65                01  FB 0001A          CALLS   #1, PAT$GET_NXT_LVT                                     ;
                                      01  DD 0001D 1$:      PUSHL   #1                                                      ; 4642
                    65                01  FB 0001F          CALLS   #1, PAT$GET_NXT_LVT                                     ;
                    52                50  D0 00022          MOVL    R0, LVT_PTR                                             ;
                                      36  13 00025          BEQL    2$                                                      ;
              04    AC          02    A2  D1 00027          CMPL    2(LVT_PTR), VALUE                                       ; 4645
                                      EF  12 0002C          BNEQ    1$                                                      ;
                    54                01  D0 0002E          MOVL    #1, ONE_FOUND                                           ; 4656
                    50                62  3C 00031          MOVZWL  (LVT_PTR), NT_PTR                                       ; 4657
                              4001    8F  BB 00034          PUSHR   #^M<R0,SP>                                              ; 4658
       00000000G    EF                02  FB 00038          CALLS   #2, PAT$ADD_NT_T_PV                                     ;
                                      5E  DD 0003F          PUSHL   SP                                                      ; 4659
       00000000G    EF                01  FB 00041          CALLS   #1, PAT$PRINT_PATH                                      ;
                                FF    A3  9F 00048          PUSHAB  -1(OUTPUT_BUFFER)                                       ; 4664
       00000000G    EF                01  FB 0004B          CALLS   #1, PAT$OUT_PUT                                         ;
                    66                53  D0 00052          MOVL    OUTPUT_BUFFER, PAT$CP_OUT_STR                           ; 4665
                         00000000G    EF  D4 00055          CLRL    PAT$GL_BUF_SIZ                                          ; 4666
                                      C0  11 0005B          BRB     1$                                                      ; 4642
                    12                54  E8 0005D 2$:      BLBS    ONE_FOUND, 3$                                           ; 4674
                                04    AC  DD 00060          PUSHL   VALUE                                                   ; 4676
                                      01  DD 00063          PUSHL   #1                                                      ;
                          006D82B8    8F  DD 00065          PUSHL   #7176888                                                ;
       00000000G    00                03  FB 0006B          CALLS   #3, LIB$SIGNAL                                          ;
                                          04 00072 3$:      RET                                                             ; 4678

; Routine Size:  115 bytes,    Routine Base:  _PAT$CODE + 0F43
```

```
GLOBAL ROUTINE PAT$REG_MATCH (STRING_DESC) =

!++

! FUNCTIONAL DESCRIPTION:
!
!       Compares a string described by the string descriptor passed as the
!       routine formal to each of the names of the machine registers.  If the
!       string matches a register name, return the number of the register (0-16,
!       where 16 is the PSL).  Otherwise return a -1.
!
! CALLING SEQUENCE:
!
!       CALLS #1, PAT$REG_MATCH
!
! INPUTS:
!
!       STRING_DESC - String descriptor to symbol string
!
! IMPLICIT INPUTS:
!
!       The VAX machine register table.
!
! OUTPUTS:
!
!       The number of the register whose name matched the string.
!
! IMPLICIT OUTPUTS:
!
!       none
!
! ROUTINE VALUE:
!
!       0-16 for the corresponding register
!       -1 for no match
!
! SIDE EFFECTS:
!
!       none
!
!--

BEGIN

MAP
        STRING_DESC : REF BLOCK [, BYTE];

LOCAL
        INDEX;

INDEX = 0;
REPEAT
        BEGIN
        IF CH$EQL (.STRING_DESC [DSC$W_LENGTH], CH$PTR (.STRING_DESC [DSC$A_POINTER]),
                        .REGISTER_TABLE [.INDEX, REG_CH_CNT],
                        CH$PTR (REGISTER_TABLE [.INDEX, REG_NAME]))
        THEN RETURN .INDEX
```

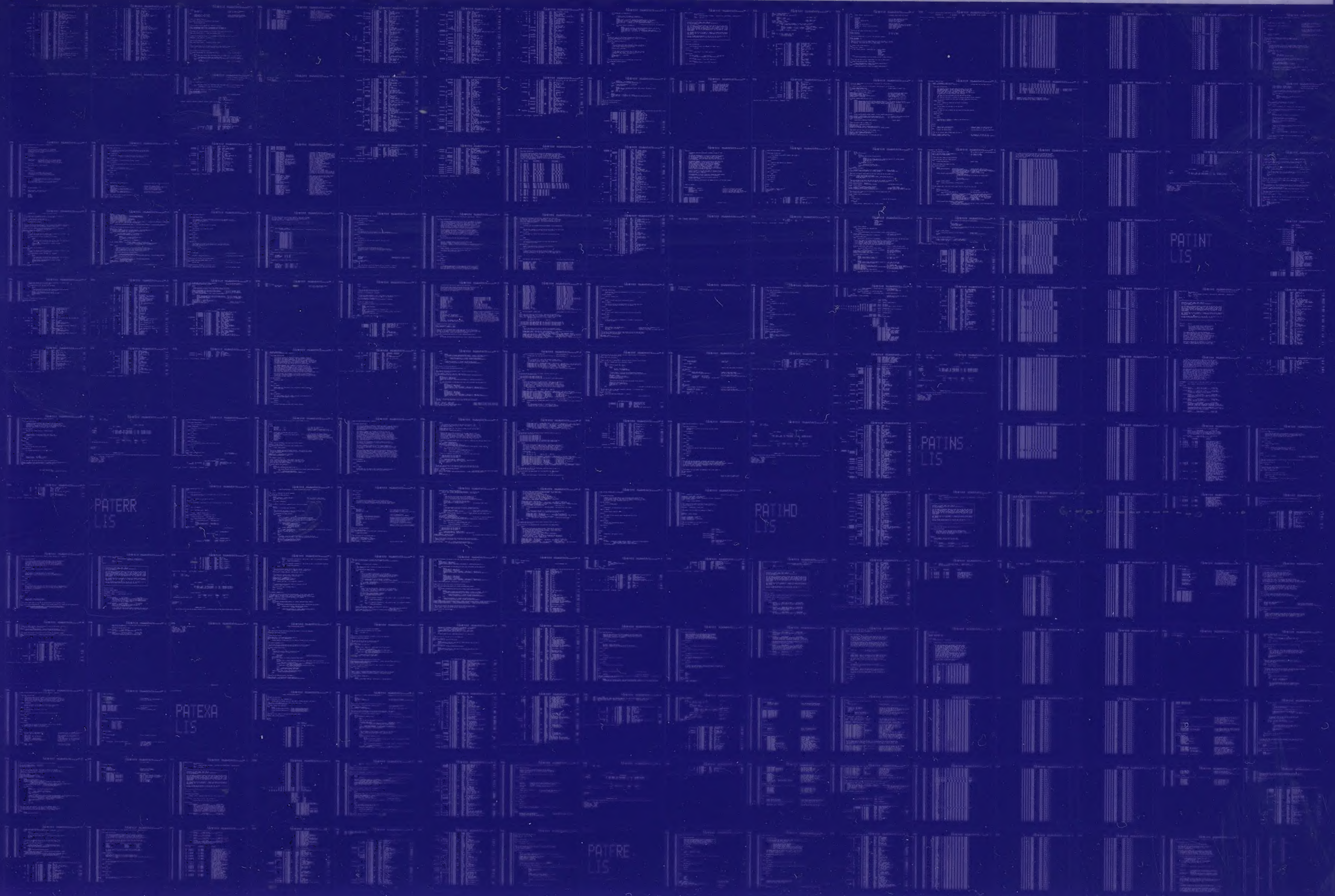
```
; 2087          4736  3          ELSE
; 2088          4737  4                  BEGIN
; 2089          4738  4                  INDEX = .INDEX + 1;
; 2090          4739  4                  IF .INDEX GTR REGISTER_COUNT - 1
; 2091          4740  4                  THEN RETURN -1
; 2092          4741  3                  END;
; 2093          4742  2          END;
; 2094          4743  1 END;
; INFO#212                L1:4729
; Null expression appears in value-required context



                                        003C 00000          .ENTRY  PAT$REG_MATCH, Save R2,R3,R4,R5              ; 4679
                                     54   D4 00002          CLRL    INDEX                                        ; 4729
                       55       04   AC   D0 00004          MOVL    STRING_DESC, R5                              ; 4732
                         00000000'EF 44   DF 00008 1$:      PUSHAL  REGISTER_TABLE[INDEX]                        ; 4733
                       50            9E   9A 0000F          MOVZBL  @(SP)+, R0                                   ;
                         00000000'EF 44   DF 00012          PUSHAL  REGISTER_TABLE+1[INDEX]                      ; 4734
        50          00        04  B5    04   BC   2D 00019          CMPC5   @STRING_DESC, @4(R5), #0, R0, @(SP)+         ;
                                     9E      00020                                                               ;
                                     04   12 00021          BNEQ    2$                                           ;
                       50            54   D0 00023          MOVL    INDEX, R0                                    ; 4735
                                          04 00026          RET                                                  ;
                                     54   D6 00027 2$:      INCL    INDEX                                        ; 4738
                       10            54   D1 00029          CMPL    INDEX, #16                                   ; 4739
                                     DA   15 0002C          BLEQ    1$                                           ;
                       50            01   CE 0002E          MNEGL   #1, R0                                       ; 4740
                                          04 00031          RET                                                  ; 4743

; Routine Size:  50 bytes,    Routine Base:  _PAT$CODE + 0FB6
```

```
1 GLOBAL ROUTINE PAT$FILL_BUF(BUF_DESC, DATA_PTR, DATA_SIZ) : NOVALUE =
1
1 !++
1 !
1 ! FUNCTIONAL DESCRIPTION:
1 !
1 !       Takes the data defined as the input arguments and puts them in the
1 !       temporary deposit buffer.  This is accomplished by allocating a new
1 !       larger buffer, copying in the old buffer, and then deallocating it.
1 !       Then the buffer descriptor is updated.
1 !
1 ! CALLING SEQUENCE:
1 !
1 !       CALLS #2, PAT$FILL_BUF
1 !
1 ! INPUTS:
1 !
1 !       BUF_DESC - Buffer descriptor
1 !       DATA_PTR - Address of the data to be put in the buffer
1 !       DATA_SIZ - Number of bytes of data to be put in the buffer
1 !
1 ! IMPLICIT INPUTS:
1 !
1 !       none
1 !
1 ! OUTPUTS:
1 !
1 !       none
1 !
1 ! IMPLICIT OUTPUTS:
1 !
1 !       The buffer descriptor is updated.
1 !
1 ! ROUTINE VALUE:
1 !
1 !       none
1 !
1 ! SIDE EFFECTS:
1 !
1 !       The data is written into the buffer.
1 !
1 !--
1
2 BEGIN
2
2 MAP
2       BUF_DESC : REF BLOCK[,BYTE];                            ! Buffer descriptor
2
2 LOCAL
2       TEMP_PTR;                                               ! Pointer to new buffer
2
2 TEMP_PTR = PAT$FREEZ((.BUF_DESC[DSC$W_LENGTH] + .DATA_SIZ + A_LONGWORD -1)/A_LONGWORD); ! Allocate larger bu
2 IF .BUF_DESC[DSC$W_LENGTH] NEQ 0
2 THEN
3       BEGIN
3       CH$MOVE(.BUF_DESC[DSC$W_LENGTH], .BUF_DESC[DSC$A_POINTER], .TEMP_PTR); ! Move in previous data
3       PAT$FREERELEASE(.BUF_DESC[DSC$A_POINTER], (.BUF_DESC[DSC$W_LENGTH] +3)/4); ! Release old buffer
```

```
; 2153          4801  2              END;
; 2154          4802  2  CH$MOVE(.DATA_SIZ, .DATA_PTR, CH$PTR(.TEMP_PTR, .BUF_DESC[DSC$W_LENGTH])); ! Move in new data
; 2155          4803  2  BUF_DESC[DSC$A_POINTER] = CH$PTR(.TEMP_PTR);                          ! Reset buffer dsc addr
; 2156          4804  2  BUF_DESC[DSC$W_LENGTH] = .BUF_DESC[DSC$W_LENGTH] + .DATA_SIZ;       ! Reset buf dsc siz
; 2157          4805  1  END;



                                         01FC 00000         .ENTRY  PAT$FILL_BUF, Save R2,R3,R4,R5,R6,R7,R8        ; 4744
                        56        04   AC  D0 00002         MOVL    BUF_DESC, R6                                     ; 4795
                        58             66  3C 00006         MOVZWL  (R6), R8                                         ;
         50             58        0C   AC  C1 00009         ADDL3   DATA_SIZ, R8, R0                                 ;
                        50             03  C0 0000E         ADDL2   #3, R0                                           ;
         7E             50             04  C7 00011         DIVL3   #4, R0, -(SP)                                    ;
               00000000G EF            01  FB 00015         CALLS   #1, PAT$FREEZ                                    ;
                        57             50  D0 0001C         MOVL    R0, TEMP_PTR                                     ;
                                       58  D5 0001F         TSTL    R8                                               ; 4796
                                       17  13 00021         BEQL    1$                                               ;
         67       04    B6             58  28 00023         MOVC3   R8, @4(R6), (TEMP_PTR)                           ; 4799
                        50        03   A8  9E 00028         MOVAB   3(R8), R0                                        ; 4800
         7E             50             04  C7 0002C         DIVL3   #4, R0, -(SP)                                    ;
                                  04   A6  DD 00030         PUSHL   4(R6)                                            ;
               00000000G EF            02  FB 00033         CALLS   #2, PAT$FREERELEASE                              ;
       6847       08    BC        0C   AC  28 0003A 1$:     MOVC3   DATA_SIZ, @DATA_PTR, (R8)[TEMP_PTR]              ; 4802
                  04    A6             57  D0 00041         MOVL    TEMP_PTR, 4(R6)                                  ; 4803
                        66        0C   AC  A0 00045         ADDW2   DATA_SIZ, (R6)                                   ; 4804
                                           04 00049         RET                                                      ; 4805

; Routine Size:  74 bytes,    Routine Base:  _PAT$CODE + 0FE8
; 2159          4806  1 END
; 2160          4807  0 ELUDOM



                                                        .EXTRN  LIB$SIGNAL

;                              PSECT SUMMARY
;
;       Name                        Bytes                         Attributes
;
;   _PAT$PLIT                          100  NOVEC,NOWRT,  RD ,NOEXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(0)
;   _PAT$CODE                         4146  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)
;   .  ABS  .                            0  NOVEC,NOWRT,NORD ,NOEXE,NOSHR,  LCL,  ABS,  CON,NOPIC,ALIGN(0)



;                         Library Statistics
;
;                                          -------- Symbols --------      Pages       Processing
;       File                               Total   Loaded   Percent      Mapped      Time
;
;   _$255$DUA28:[SYSLIB]LIB.L32;1          18619       7         0       1000          00:01.8



; Information:  1
; Warnings:     0
; Errors:       0



;                              COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/VARIANT:1/LIS=LIS$:PATEXA/OBJ=OBJ$:PATEXA MSRC$:PATEXA/UPDATE=(ENH$:PATEXA)

; Size:          4146 code + 100 data bytes
; Run Time:         01:20.5
; Elapsed Time:     04:10.7
; Lines/CPU Min:     3581
; Lexemes/CPU-Min: 28636
; Memory Used:  406 pages
; Compilation Complete
```

0301 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY


PATINT
LIS

PATINS
LIS

PATERR
LIS

PATIHD
LIS

PATEXA
LIS

PATFRE
LIS